- AKTU 1st Year Sem 1 Solved Paper 2016-17 | COMP. SYSTEM & C PROGRAMMING | Sec A

Paper download link: Paper | Sem 1 | 2016-17 (https://media.geeksforgeeks.org/wp-content/uploads/Paper-Sem-1-2016-17.pdf)

B.Tech.
(SEM-II) THEORY EXAMINATION 2016-17
COMPUTER SYSTEM & PROGRAMMING IN C

Time: 3hrs
Total Marks: 100

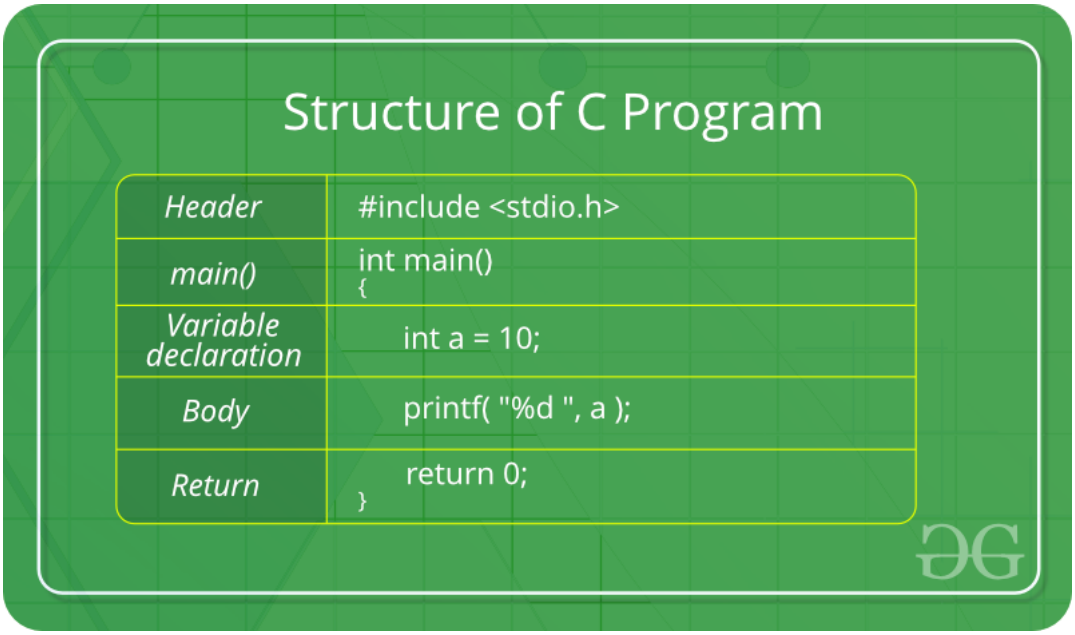
Note:-

- There are **three** sections. Section **A** carries **20** marks, Section **B** carries **30** marks and Section **C** carries **50** marks.
- Attempt all questions. Marks are indicated against each question.
- Assume suitable data wherever necessary.

Section – A

1. Attempt all questions: (2*10 = 20)

a. **Explain the basic structure of a C program (https://www.geeksforgeeks.org/c-language-set-1-introduction/).**
The structure of a C program is as follows:



1. Header Files Inclusion
2. Main Method Declaration
3. Variable Declaration
4. Body
5. Return Statement

b. **What do you mean by algorithm (https://www.geeksforgeeks.org/fundamentals-of-algorithms/)? Explain the characteristics of algorithm.**
The word **Algorithm (https://www.geeksforgeeks.org/fundamentals-of-algorithms/)** means "a process or set of rules to be followed in calculations or other problem-solving operations". Therefore Algorithm refers to a set of rules/instructions that step-by-step define how a work is to be executed upon inorder to get the expected results.

Characteristics of an Algorithm

- **Clear and Unambiguous**: Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs**: If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- **Finiteness:** The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.
- **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon will the available resources. It must not contain some future technology, or anything.
- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

c. **What are functions (https://www.geeksforgeeks.org/functions-in-c/)? What is the advantage of using multiple functions in a program?**
A function is a set of statements that take inputs, do some specific computation and produces output.

The idea is to put some commonly or repeatedly done task together and make a function, so that instead of writing the same code again and again for different inputs, we can call the function.

Advantage of using multiple functions in a program

1. **Ease of Use :**This approach allows simplicity, as rather than focusing on the entire thousands and millions of lines code in one go we can access it in the form of modules. This allows ease in debugging the code and prone to less error.
2. **Reusability :**It allows the user to reuse the functionality with a different interface without typing the whole program again.
3. **Ease of Maintenance :** It helps in less collision at the time of working on modules, helping a team to work with proper collaboration while working on a large application.

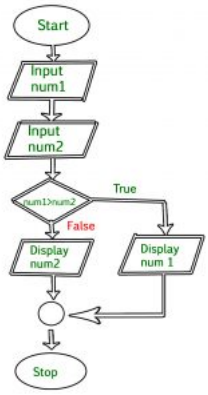
d. **Distinguish between int main() and void main()?**
int main(): This prototype refers to the main function in a C program that returns an integer value. This integer value is the exit code of the program that defines if the program has completed successfully or not. For successful execution, 0 is returned. Else any other value is returned. This format is now the standard ANSI defined format for the main() method in C programming.

void main(): This prototype refers to the main function in a C program that do not returns any value. Earlier this prototype were used but now this format is not recommended by the standards and must not be used.

e. **What is the difference between pseudo-code (https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/) and flowchart (https://www.geeksforgeeks.org/an-introduction-to-flowcharts/)?**
Pseudo-code is a step-by-step representation of how to solve a problem in the form of **text** in any language.
For example:

```
check whether the number is even or odd.

if "1"
    print response
        "I am case 1"

if "2"
    print response
        "I am case 2
```

Flowchart is a step-by-step representation of how to solve a problem in the form of **graphical representation** For example: Flowchart to input two numbers from user and display the largest of two numbers



f. **Draw the memory hierarchical structure of a computer system (https://www.geeksforgeeks.org/memory-hierarchy-design-and-its-characteristics/).**

MEMORY HIERARCHY DESIGN

g. **What will be the output of following code?**

```
1
2  void main()
3  {
4      int a = 5, b = 6;
5      printf("%d\t", a = b);
6      printf("%d\t", a == b);
7      printf("%d\t%d", a, b);
8  }
9
```

Output:

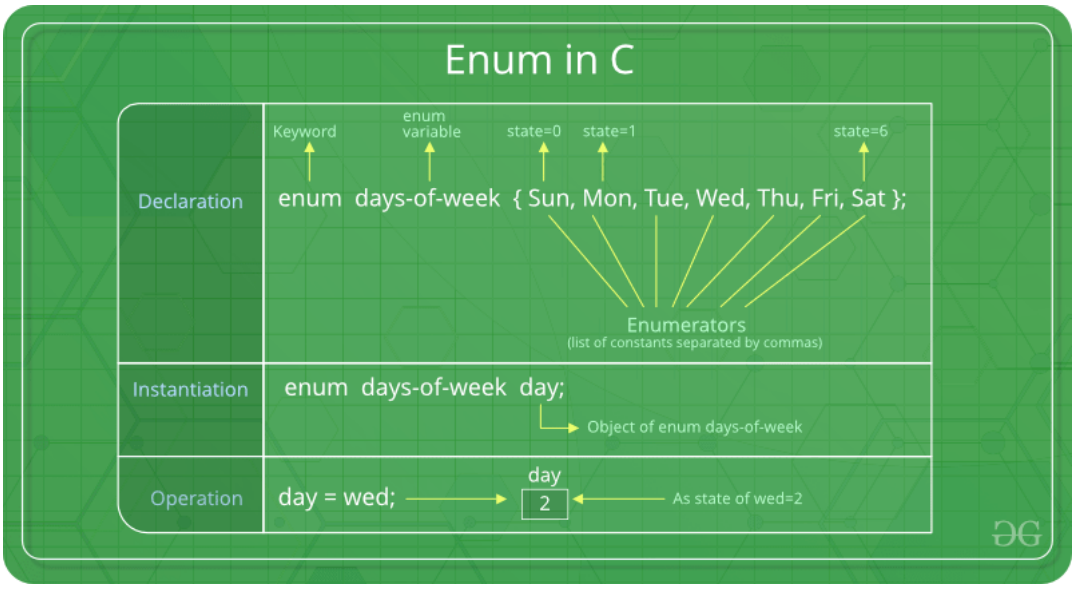
```
6    1    6    6
```

h. **Write short notes on High level and low level languages (https://www.geeksforgeeks.org/introduction-to-programming-languages/).**
High level languages are nearly human languages. In other words, these are the languages in which the code are written in a

language which are more understood by the human than by the machines.

Low level languages are nearly machine languages. In other words, these are the languages in which the code are written in a language which are more understood by the machines than by the humans.

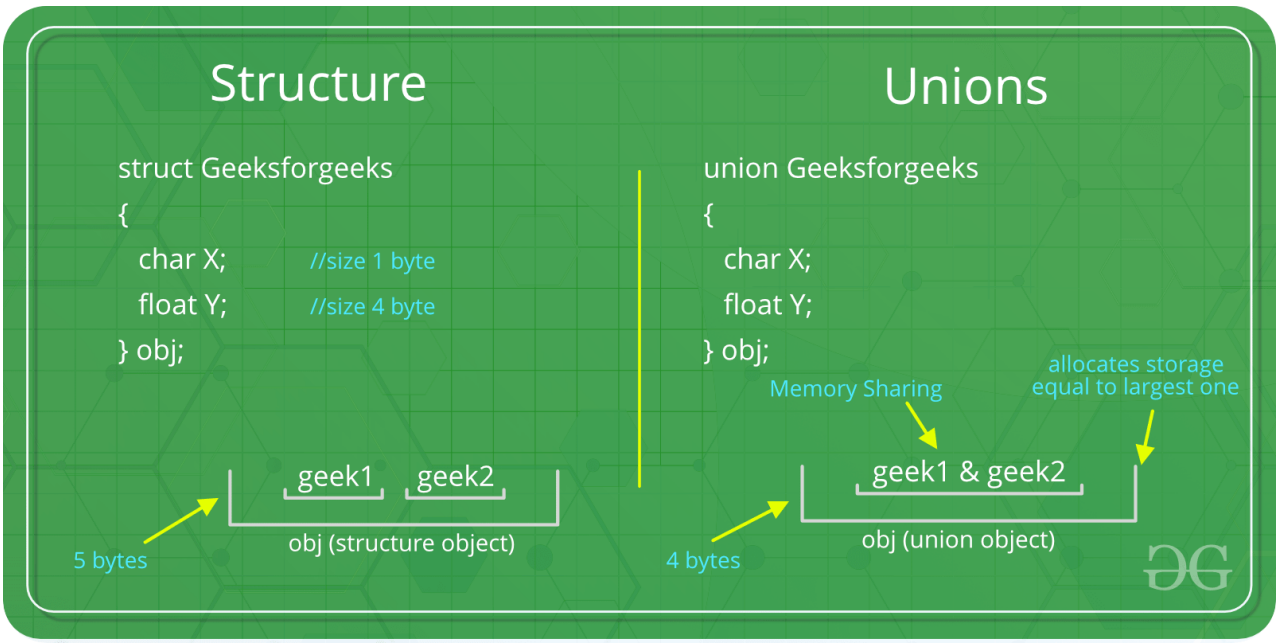
i. **Write short note on Union (https://www.geeksforgeeks.org/union-c/) and enumerated data type (https://www.geeksforgeeks.org/enumeration-enum-c/).**
Enumerated Data-type: Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.




```
enum State {Working = 1, Failed = 0};
```

The keyword 'enum' is used to declare new enumeration types in C and C++.

Union: Like Structures (https://www.geeksforgeeks.org/structures-c/), union is a user defined data type. In union, all members share the same memory location.



j. **Write five commands of LINUX with its architecture?**
1. **who (https://www.geeksforgeeks.org/who-command-in-linux/)**

```
$who [options] [filename]
```

2. ping (https://www.geeksforgeeks.org/ping-command-in-linux-with-examples/)

```
sudo ping -v
```

3. tar (https://www.geeksforgeeks.org/tar-command-linux-examples/)

```
tar [options] [archive-file] [file or directory to be archived]
```

4. netstat (https://www.geeksforgeeks.org/netstat-command-linux/)

```
# netstat -a | more
```

5. expand (https://www.geeksforgeeks.org/expand-command-in-linux-with-examples/)

```
$expand [OPTION] FILE
```


- AKTU 1st Year Sem 1 Solved Paper 2016-17 | COMP. SYSTEM & C PROGRAMMING | Sec B

Paper download link: Paper | Sem 1 | 2016-17 (https://media.geeksforgeeks.org/wp-content/uploads/Paper-Sem-1-2016-17.pdf)

B.Tech.
(SEM-II) THEORY EXAMINATION 2016-17
COMPUTER SYSTEM & PROGRAMMING IN C

Time: 3hrs
Total Marks: 100

Note:-

- There are **three** sections. Section **A** carries **20** marks, Section **B** carries **30** marks and Section **C** carries **50** marks.
- Attempt all questions. Marks are indicated against each question.
- Assume suitable data wherever necessary.

Section – B

2. Attempt any five questions: (5*10 = 50)

a) i) What is the role of SWITCH statement in C programming language (https://www.geeksforgeeks.org/switch-statement-cc/). Explain with example.

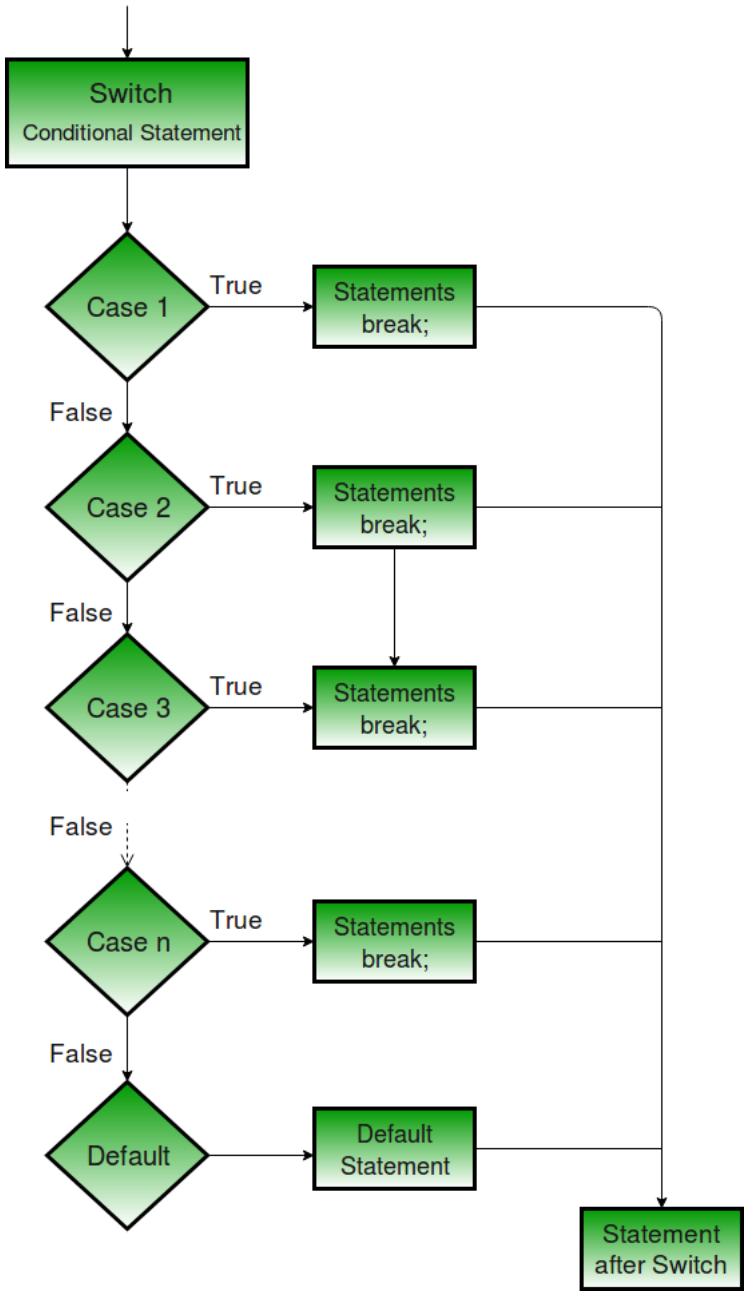
Switch case statements are a substitute for long if statements that compare a variable to several integral values

- The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.
- Switch is a control statement that allows a value to change control of execution.

Syntax:

```
switch (n)

{

    case 1: // code to be executed if n = 1;

        break;

    case 2: // code to be executed if n = 2;

        break;

    default: // code to be executed if n doesn't match any cases

}
```

Flowchart:



Example:

```
1
2  // Following is a simple program to demonstrate
3  // syntax of switch.
4  #include <stdio.h>
5  int main()
6  {
7      int x = 2;
8      switch (x) {
```

```
    switch (x) {
    case 1:
        printf("Choice is 1");
        break;
    case 2:
        printf("Choice is 2");
        break;
    case 3:
        printf("Choice is 3");
        break;
    default:
        printf("Choice other than 1, 2 and 3");
        break;
    }
    return 0;
}

```

Run

Output:

```
Choice is 2
```

a) ii) What is recursion (http://www.geeksforgeeks.org/recursion/)? Write a program in C to generate the Fibonacci series (https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/).

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples of such problems are Towers of Hanoi (TOH) (http://quiz.geeksforgeeks.org/c-program-for-tower-of-hanoi/), Inorder/Preorder/Postorder Tree Traversals (https://www.cdn.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/), DFS of Graph (https://www.cdn.geeksforgeeks.org/depth-first-traversal-for-a-graph/), etc.

Base condition in recursion? In recursive program, the solution to base case is provided and solution of bigger problem is expressed in terms of smaller problems.

```
int fact(int n)
{
    if (n < = 1) // base case
        return 1;
    else
        return n*fact(n-1);
}
```

In the above example, base case for n < = 1 is defined and larger value of number can be solved by converting to smaller one till base case is reached.

Program in C to generate the Fibonacci series (https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/):

```

// Fibonacci Series using Recursion

#include <stdio.h>

int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n - 1) + fib(n - 2);
}

int main()
{
    int n = 9;
    printf("%d", fib(n));
    getchar();
    return 0;
}

```

Run

Output:

34

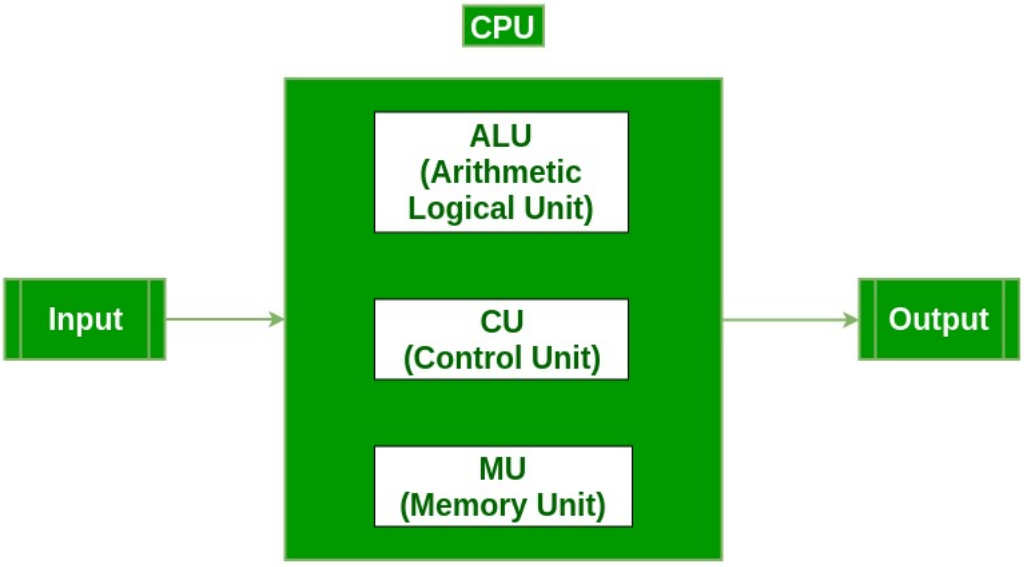
b) i) Differentiate between-

1. Actual and formal parameters (https://www.geeksforgeeks.org/parameter-passing-techniques-in-c-cpp/):
   - **Formal Parameter :** A variable and its type as they appear in the prototype of the function or method.
   - **Actual Parameter :** The variable or expression corresponding to a formal parameter that appears in the function or method call in the calling environment.
2. global (https://www.geeksforgeeks.org/scope-of-variables-in-c/) and extern (https://www.geeksforgeeks.org/understanding-extern-keyword-in-c/) variables
   - As the name suggests, Global Variables can be accessed from any part of the program.
     - They are available through out the life time of a program.
     - They are declared at the top of the program outside all of the functions or blocks.
     - **Declaring global variables**: Global variables are usually declared outside of all of the functions and blocks, at the top of the program. They can be accessed from any portion of the program.
   - extern: Extern storage class simply tells us that the variable is defined elsewhere and not within the same block where it is used. Basically, the value is assigned to it in a different block and this can be overwritten/changed in a different block as well. So an extern variable is nothing but a global variable initialized with a legal value where it is declared in order to be used elsewhere. It can be accessed within any function/block. Also, a normal global variable can be made extern as well by placing the 'extern' keyword before its declaration/definition in any function/block. This basically signifies that we are not initializing a new variable but instead we are using/accessing the global variable only. The main purpose of using extern variables is that they can be accessed between two different files which are part of a large program. For more information on how extern variables work, have a look at this link.

b) ii) Describe about the basic components of a computer (https://www.geeksforgeeks.org/functional-components-of-a-computer/) with a neat block diagram.

Digital Computer: A digital computer can be defined as a programmable machine which reads the binary data passed as instructions, processes this binary data, and displays a calculated digital output. Therefore, Digital computers are those that work on the digital data.

Details of Functional Components of a Digital Computer



- **Input Unit :**The input unit consists of input devices that are attached to the computer. These devices take input and convert it into binary language that the computer understands. Some of the common input devices are keyboard, mouse, joystick, scanner etc.
- **Central Processing Unit (CPU0 :** Once the information is entered into the computer by the input device, the processor processes it. The CPU is called the brain of the computer because it is the control center of the computer. It first fetches instructions from memory and then interprets them so as to know what is to be done. If required, data is fetched from memory or input device. Thereafter CPU executes or performs the required computation and then either stores the output or displays on the output device. The CPU has three main components which are responsible for different functions – Arithmetic Logic Unit (ALU), Control Unit (CU) and Memory registers
- **Arithmetic and Logic Unit (ALU) :** The ALU, as its name suggests performs mathematical calculations and takes logical decisions. Arithmetic calculations include addition, subtraction, multiplication and division. Logical decisions involve comparison of two data

items to see which one is larger or smaller or equal.

- **Control Unit :** The Control unit coordinates and controls the data flow in and out of CPU and also controls all the operations of ALU, memory registers and also input/output units. It is also responsible for carrying out all the instructions stored in the program. It decodes the fetched instruction, interprets it and sends control signals to input/output devices until the required operation is done properly by ALU and memory.
- **Memory Registers :** A register is a temporary unit of memory in the CPU. These are used to store the data which is directly used by the processor. Registers can be of different sizes(16 bit, 32 bit, 64 bit and so on) and each register inside the CPU has a specific function like storing data, storing an instruction, storing address of a location in memory etc. The user registers can be used by an assembly language programmer for storing operands, intermediate results etc. Accumulator (ACC) is the main register in the ALU and contains one of the operands of an operation to be performed in the ALU.
- **Memory (https://www.geeksforgeeks.org/types-computer-memory-ram-rom/) :** Memory attached to the CPU is used for storage of data and instructions and is called internal memory The internal memory is divided into many storage locations, each of which can store data or instructions. Each memory location is of the same size and has an address. With the help of the address, the computer can read any memory location easily without having to search the entire memory. when a program is executed, it's data is copied to the internal memory ans is stored in the memory till the end of the execution. The internal memory is also called the Primary memory or Main memory. This memory is also called as RAM, i.e. Random Access Memory. The time of access of data is independent of its location in memory, therefore this memory is also called Random Access memory (RAM). Read this for different types of RAMs (https://www.geeksforgeeks.org/different-types-ram-random-access-memory/)
- **Output Unit :** The output unit consists of output devices that are attached with the computer. It converts the binary data coming from CPU to human understandable form. The common output devices are monitor, printer, plotter etc.

c) i) What do you mean by dynamic memory allocation (https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/)? Explain the malloc() and calloc() function in detail.
Dynamic Memory Allocation in C: It can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime.

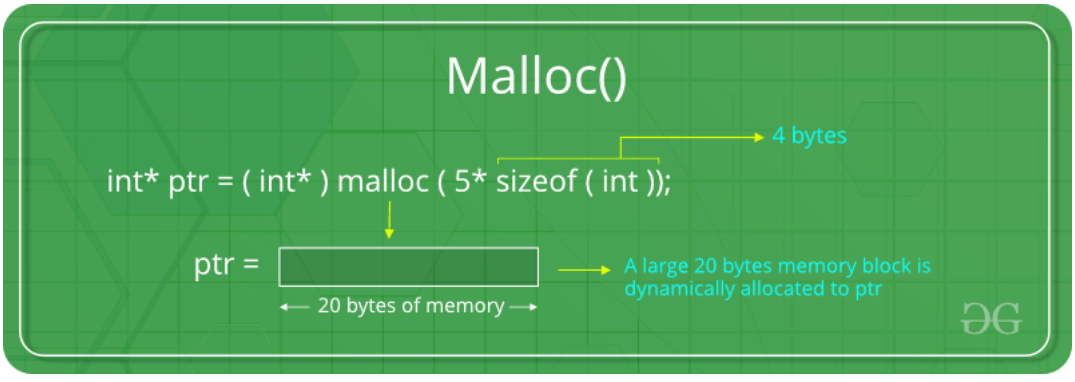
C provides some functions to achieve these tasks. There are 4 library functions provided by C defined under **<stdlib.h>** header file to facilitate dynamic memory allocation in C programming. They are:

1. malloc()
2. calloc()
3. free()
4. realloc()

1. **malloc()**
   "malloc" or **"memory allocation"** method is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form.

   Syntax:

```
ptr = (cast-type*) malloc(byte-size)

For Example:
ptr = (int*) malloc(100 * sizeof(int));

Since the size of int is 4 bytes,
this statement will allocate 400 bytes of memory.
And, the pointer ptr holds the address
of the first byte in the allocated memory.
```



If the space is insufficient, allocation fails and returns a NULL pointer.

Example:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{

    // This pointer will hold the
    // base address of the block created
    int* ptr;
    int n, i, sum = 0;

    // Get the number of elements for the array
    n = 5;
    printf("Enter number of elements: %d\n", n);

    // Dynamically allocate memory using malloc()
    ptr = (int*)malloc(n * sizeof(int));

    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {

        // Memory has been successfully allocated
        printf("Memory successfully allocated using malloc.\n");

```

Run

Output:

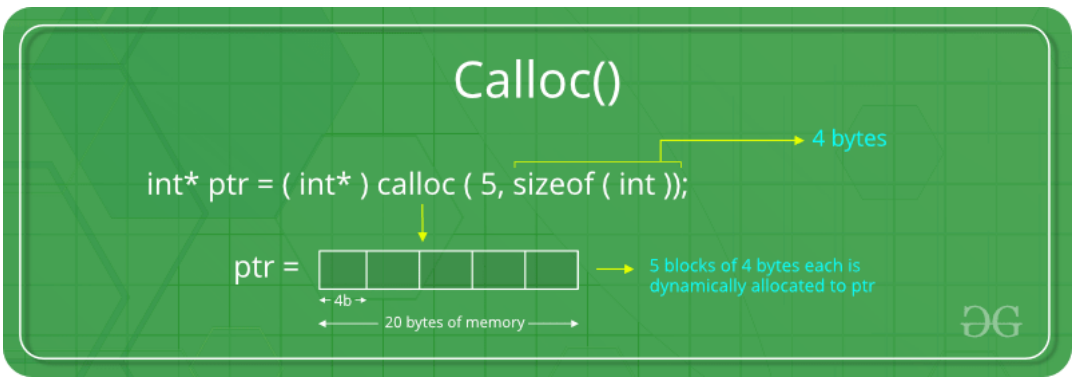
```
Enter number of elements: 5
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5,
```

2. calloc()

"calloc" or "contiguous allocation" method is used to dynamically allocate the specified number of blocks of memory of the specified type. It initializes each block with a default value '0'.

Syntax:

```
ptr = (cast-type*)calloc(n, element-size);

For Example:
ptr = (float*) calloc(25, sizeof(float));

This statement allocates contiguous space in memory
for 25 elements each with the size of float.
```



If the space is insufficient, allocation fails and returns a NULL pointer.

Example:

```c

#include <stdio.h>
#include <stdlib.h>

int main()
{

    // This pointer will hold the
    // base address of the block created
    int* ptr;
    int n, i, sum = 0;

    // Get the number of elements for the array
    n = 5;
    printf("Enter number of elements: %d\n", n);

    // Dynamically allocate memory using calloc()
    ptr = (int*)calloc(n, sizeof(int));

    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {

        // Memory has been successfully allocated
        printf("Memory successfully allocated using calloc.\n");

```

Run

Output:

```
Enter number of elements: 5
Memory successfully allocated using calloc.
The elements of the array are: 1, 2, 3, 4, 5,
```

c) ii) Write a program to multiply two matrices of dimension 3*3 and store the result in another matrix.

```c

// C program to multiply two square matrices.

#include <stdio.h>

const int MAX = 100;

// Function to print Matrix
void printMatrix(int M[][MAX], int rowSize, int colSize)
{
    for (int i = 0; i < rowSize; i++) {
        for (int j = 0; j < colSize; j++)
            printf("%d ", M[i][j]);

        printf("\n");
    }
}

// Function to multiply two matrices A[][] and B[][]
void multiplyMatrix(int row1, int col1, int A[][MAX],
                    int row2, int col2, int B[][MAX])
{
    int i, j, k;

    // Matrix to store the result
    int C[MAX][MAX];

    // Check if multiplication is Possible
    if (row2 != col1) {
        printf("Not Possible\n");
```

Run

Output:

```
Enter the number of rows of First Matrix: 2
Enter the number of columns of First Matrix: 3
Enter the elements of First Matrix:
A[0][0]: 1
A[0][1]: 2
A[0][2]: 3
A[1][0]: 4
A[1][1]: 5
A[1][2]: 6

Enter the number of rows of Second Matrix: 3
Enter the number of columns of Second Matrix: 2
Enter the elements of First Matrix:
B[0][0]: 1
B[0][1]: 2
B[1][0]: 3
B[1][1]: 4
B[2][0]: 5
B[2][1]: 6

First Matrix:
1 2 3
4 5 6

Second Matrix:
1 2
3 4
5 6

Resultant Matrix:
22 28
49 64
```


d) Convert the following numbers: (i) $(10101011101.011)_2 = ()_{16}$

- $(10101011101.011)_2 = (55D.6)_{16}$

(ii) $(916.125)_{10} = ()_4$

- $(916.125)_{10} = (32110.02)_4$

(iii) $(123)_{10} = ()_2$

- $(123)_{10} = (1111011)_2$

(iv) $(574.32)_8 = ()_2$

- $(574.32)_8 = (101111100.01101)_2$

(v) $(1011.10)_2 = ()_{10}$

- $(1011.10)_2$

$= (11.5)_{10}$

e) i) Write a program to print all prime numbers (https://www.geeksforgeeks.org/c-program-to-check-whether-a-number-is-prime-or-not/) from 1 to 300.

```c

// C program to check if a
// number is prime

#include <stdio.h>

int main()
{
    int n = 1, i, flag = 1;

    for (n = 1; n <= 300; n++) {
        flag = 1;
        // Iterate from 2 to n/2
        for (i = 2; i <= n / 2; i++) {
```

```
14          for (i = 2; i <= n / 2; i++) {
15
16              // If n is divisible by any number between
17              // 2 and n/2, it is not prime
18              if (n % i == 0) {
19                  flag = 0;
20                  break;
21              }
22          }
23
24          if (flag == 1) {
25              printf("\n%d is a prime number", n);
26          }
27      }
28      return 0;
29  }
30
```


Run

Output:

```
1 is a prime number
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
43 is a prime number
47 is a prime number
53 is a prime number
59 is a prime number
61 is a prime number
67 is a prime number
71 is a prime number
73 is a prime number
79 is a prime number
83 is a prime number
89 is a prime number
97 is a prime number
101 is a prime number
103 is a prime number
107 is a prime number
109 is a prime number
113 is a prime number
127 is a prime number
131 is a prime number
137 is a prime number
139 is a prime number
149 is a prime number
151 is a prime number
157 is a prime number
163 is a prime number
167 is a prime number
173 is a prime number
179 is a prime number
181 is a prime number
191 is a prime number
193 is a prime number
197 is a prime number
199 is a prime number
211 is a prime number
223 is a prime number
227 is a prime number
229 is a prime number
233 is a prime number
239 is a prime number
241 is a prime number
251 is a prime number
257 is a prime number
263 is a prime number
269 is a prime number
271 is a prime number
277 is a prime number
281 is a prime number
283 is a prime number
293 is a prime number
```


e) ii) Any year is input through the keyboard. Write a program to determine whether the year is a leap year or not (https://www.geeksforgeeks.org/program-check-given-year-leap-year/).

```
1
2  // C program to check if a given
3  // year is leap year or not
4
5  #include <stdbool.h>
6  #include <stdio.h>
7
```

```
bool checkYear(int year)
{
    // If a year is multiple of 400,
    // then it is a leap year
    if (year % 400 == 0)
        return true;

    // Else If a year is multiple of 100,
    // then it is not a leap year
    if (year % 100 == 0)
        return false;

    // Else If a year is multiple of 4,
    // then it is a leap year
    if (year % 4 == 0)
        return true;
    return false;
}

// driver code
int main()
{
    int year;
```


Run

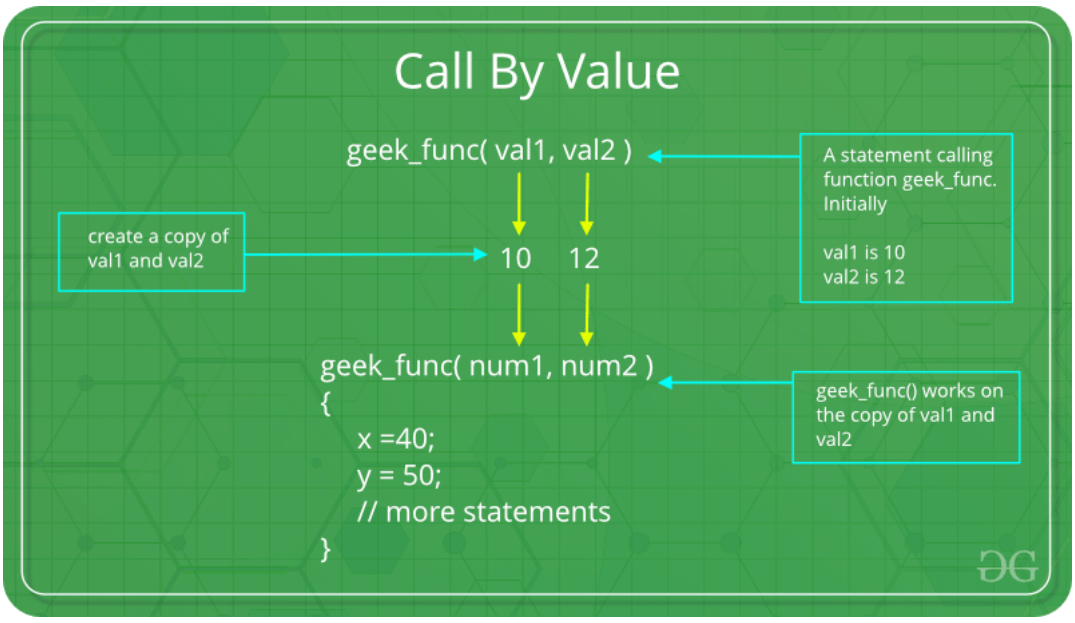
Output:

```
Enter the year: 2018
Not a Leap Year
```

f) What do you mean by parameter passing (https://www.geeksforgeeks.org/parameter-passing-techniques-in-c-cpp/)? Discuss various types of parameter passing mechanism in C with examples (https://www.geeksforgeeks.org/parameter-passing-techniques-in-c-cpp/).
Assigning values to the variables defined in the function definition in runtime is known as parameter passing. There are different ways in which parameter data can be passed into and out of methods and functions. Let us assume that a function *B()* is called from another function *A()*. In this case *A* is called the ***"caller function"*** and *B* is called the ***"called function or callee function"***. Also, the arguments which *A* sends to *B* are called *actual arguments* and the parameters of *B* are called *formal arguments*.

Important methods of Parameter Passing

1. **Pass By Value :** This method uses *in-mode* semantics. Changes made to formal parameter do not get transmitted back to the caller. Any modifications to the formal parameter variable inside the called function or method affect only the separate storage location and will not be reflected in the actual parameter in the calling environment. This method is also called as ***call by value***.




```
// C program to illustrate
// call by value
#include <stdio.h>

void func(int a, int b)
{
    a += b;
```

```
    printf("In func, a = %d b = %d\n", a, b);
}
int main(void)
{
    int x = 5, y = 7;

    // Passing parameters
    func(x, y);
    printf("In main, x = %d y = %d\n", x, y);
    return 0;
}
```

Run

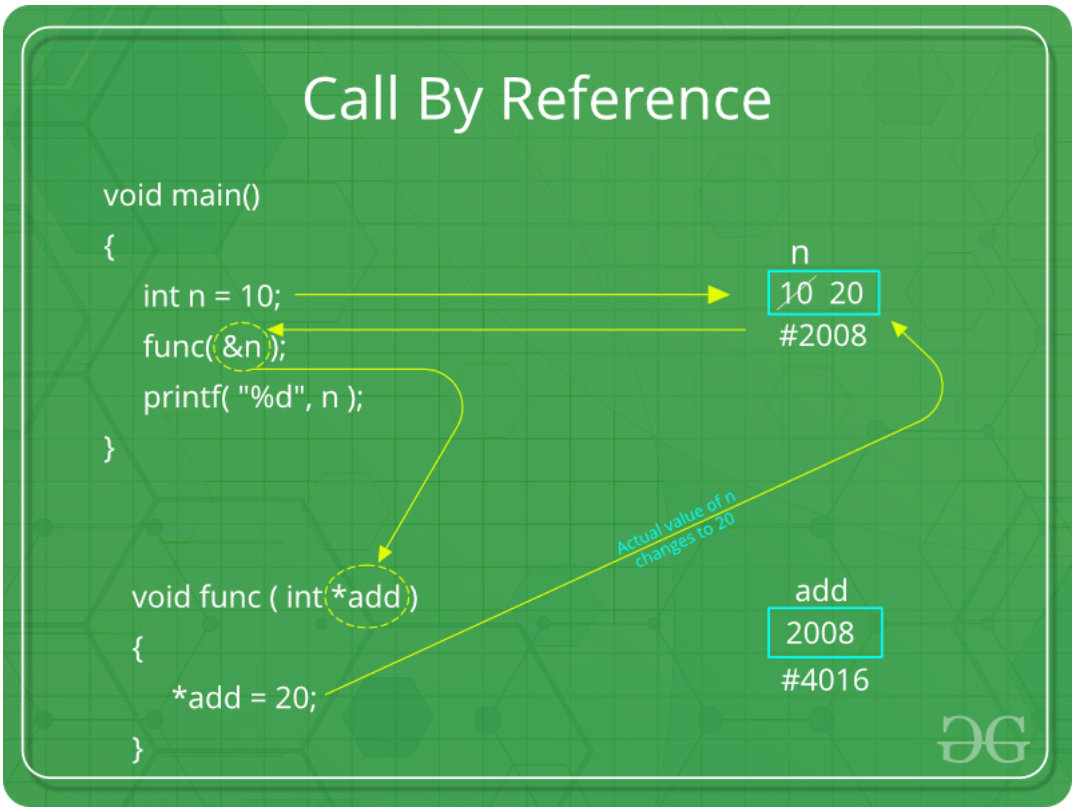
Output:

```
In func, a = 12 b = 7
In main, x = 5 y = 7
```

Languages like C, C++, Java support this type of parameter passing. Java in fact is strictly call by value. (https://www.geeksforgeeks.org/g-fact-31-java-is-strictly-pass-by-value/) **Shortcomings:**

- Inefficiency in storage allocation
- For objects and arrays, the copy semantics are costly

2. **Pass by reference(aliasing) :** This technique uses *in/out-mode* semantics. Changes made to formal parameter do get transmitted back to the caller through parameter passing. Any changes to the formal parameter are reflected in the actual parameter in the calling environment as formal parameter receives a reference (or pointer) to the actual data. This method is also called as call by reference. This method is efficient in both time and space.




```
// C program to illustrate
// call by reference
#include <stdio.h>

void swapnum(int* i, int* j)
{
    int temp = *i;
    *i = *j;
    *j = temp;
```

```
    *j = temp;
}

int main(void)
{
    int a = 10, b = 20;

    // passing parameters
    swapnum(&a, &b);

    printf("a is %d and b is %d\n", a, b);
    return 0;
}

```


Run

Output:

```
a is 20 and b is 10
```

g) i) Define data types in C (https://www.geeksforgeeks.org/data-types-in-c/). Discuss primitive data types in terms of memory occupied, format specifier and range.

Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it. Let us briefly describe them one by one:

Following are the examples of some very common data types used in C:

- **char:** The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.
- **int:** As the name suggests, an int variable is used to store an integer.
- **float:** It is used to store decimal numbers (numbers with floating point value) with single precision.
- **double:** It is used to store decimal numbers (numbers with floating point value) with double precision.

Different data types also have different ranges upto which they can store numbers. These ranges may vary from compiler to compiler. Below is list of ranges along with the memory requirement and format specifiers on 32 bit gcc compiler.

Data Type	Memory (bytes)	Range	Format Specifier
short int	2	-32, 768 to 32, 767	%hd
unsigned short int	2	0 to 65, 535	%hu
unsigned int	4	0 to 4, 294, 967, 295	%u
int	4	-2, 147, 483, 648 to 2, 147, 483, 647	%d
long int	4	-2, 147, 483, 648 to 2, 147, 483, 647	%ld
unsigned long int	4	0 to 4, 294, 967, 295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18, 446, 744, 073, 709, 551, 615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	12		%Lf

g) ii) Write a program to print the following pattern:

```
A
AB
ABC
ABCD
ABCDE
ABCDEF
```



```
#include <stdio.h>

int main()
{

    char ch = 'A';
    int i, j;

    for (i = 0; i < 5; i++) {
        for (j = 0; j <= i; j++) {
            printf("%c", ch + j);
        }

        printf("\n");
    }

    return 0;
}

```

Run

Output:

```
A
AB
ABC
ABCD
ABCDE
```

h) List out various file operations and modes in C. Write a program to copy the content from one file to another file. File opening modes in C: (https://www.geeksforgeeks.org/basics-file-handling-c/)

- **“r” -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.
- **“w” -** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- **"a" -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- **"r+" -** Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.
- **"w+" -** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file.
- **"a+" -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

C program to copy contents of one file to another file:

```
#include <stdio.h>
#include <stdlib.h> // For exit()

int main()
{
    FILE *fptr1, *fptr2;
    char filename[100], c;

    printf("Enter the filename to open for reading \n");
```

```
    scanf("%s", filename);

    // Open one file for reading
    fptr1 = fopen(filename, "r");
    if (fptr1 == NULL) {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }

    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);

    // Open another file for writing
    fptr2 = fopen(filename, "w");
    if (fptr2 == NULL) {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }

    // Read contents from file
```

Run

Output:

```
Enter the filename to open for reading
Cannot open file
```

Output:

```
Enter the filename to open for reading
a.txt
Enter the filename to open for writing
b.txt
Contents copied to b.txt
```


- AKTU 1st Year Sem 1 Solved Paper 2016-17 | COMP. SYSTEM & C PROGRAMMING | Sec C

Paper download link: Paper | Sem 1 | 2016-17 (https://media.geeksforgeeks.org/wp-content/uploads/Paper-Sem-1-2016-17.pdf)

B.Tech.
(SEM-II) THEORY EXAMINATION 2016-17
COMPUTER SYSTEM & PROGRAMMING IN C

Time: 3hrs
Total Marks: 100

Note:-

- There are **three** sections. Section **A** carries **20** marks, Section **B** carries **30** marks and Section **C** carries **50** marks.
- Attempt all questions. Marks are indicated against each question.
- Assume suitable data wherever necessary.

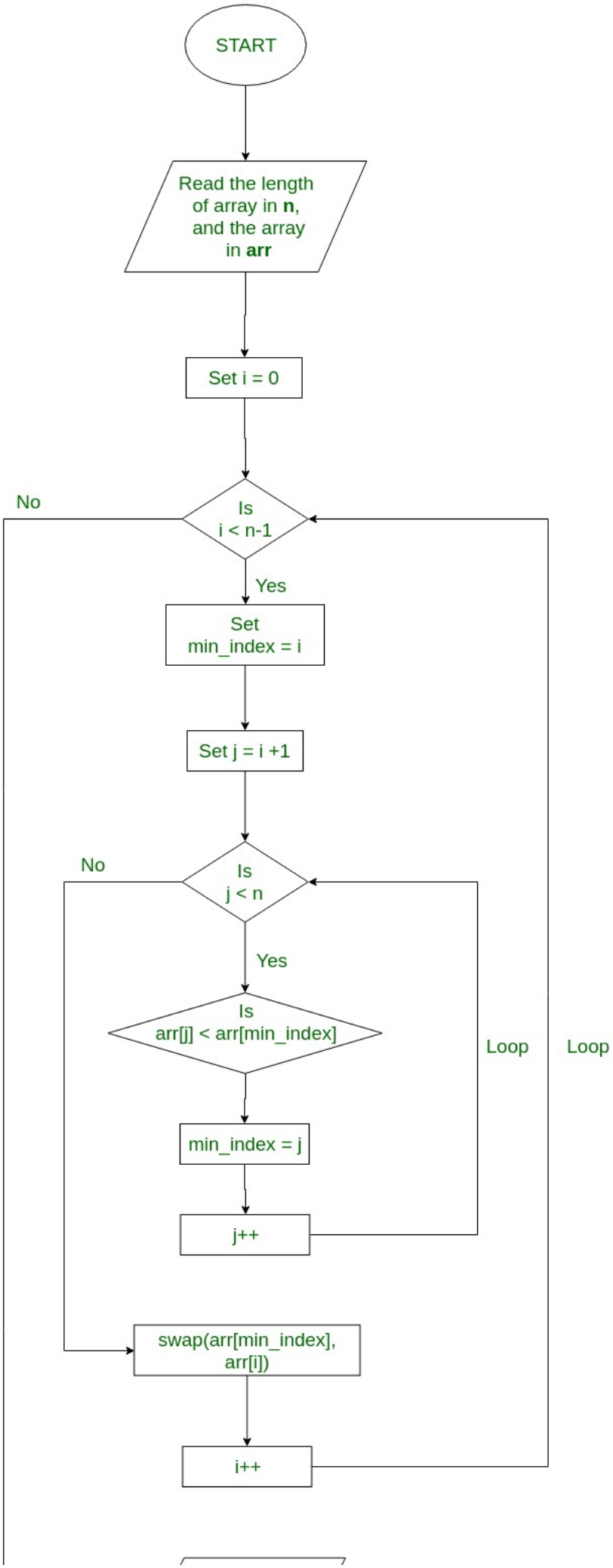
Section – C

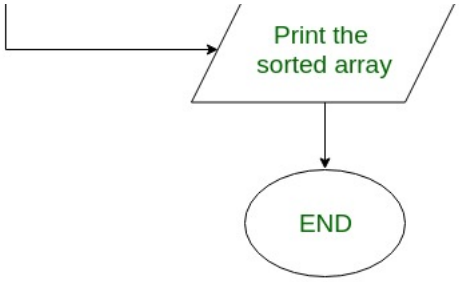
Attempt any two questions from this section: (2*15 = 30)

3. a) What do you mean by sorting. Write a program in C to sort the given n positive integers. Also give the flowchart for the same.
A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.

For example: The below list of characters is sorted in increasing order of their ASCII values. That is, the character with lesser ASCII value will be placed first than the character with higher ASCII value.

Flowchart of the Selection Sort:





Flowchart for Selection Sort

Program in C to sort the elements of a given array of N positive integers:

```c

// C program to sort the elements
// of a given array of N positive integers

#include <stdio.h>

void swap(int* xp, int* yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n - 1; i++) {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i + 1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}

```

Run

Output:

```
Sorted array:
11 12 22 25 64
```

3. b) Write a program to check whether a given number is Armstrong (https://www.geeksforgeeks.org/program-for-armstrong-numbers/) or not. Like 153 = 13 + 53 + 33.

```c

// C program to find Armstrong number

#include <stdio.h>

/* Function to calculate x raised to the power y */
int power(int x, unsigned int y)
{
    if (y == 0)
        return 1;
    if (y % 2 == 0)
        return power(x, y / 2) * power(x, y / 2);
    return x * power(x, y / 2) * power(x, y / 2);
}

/* Function to calculate order of the number */
int order(int x)
{
    int n = 0;
    while (x) {
```

```
        n++;
        x = x / 10;
    }
    return n;
}

// Function to check whether the given number is
// Armstrong number or not
int isArmstrong(int x)
{
```

Run

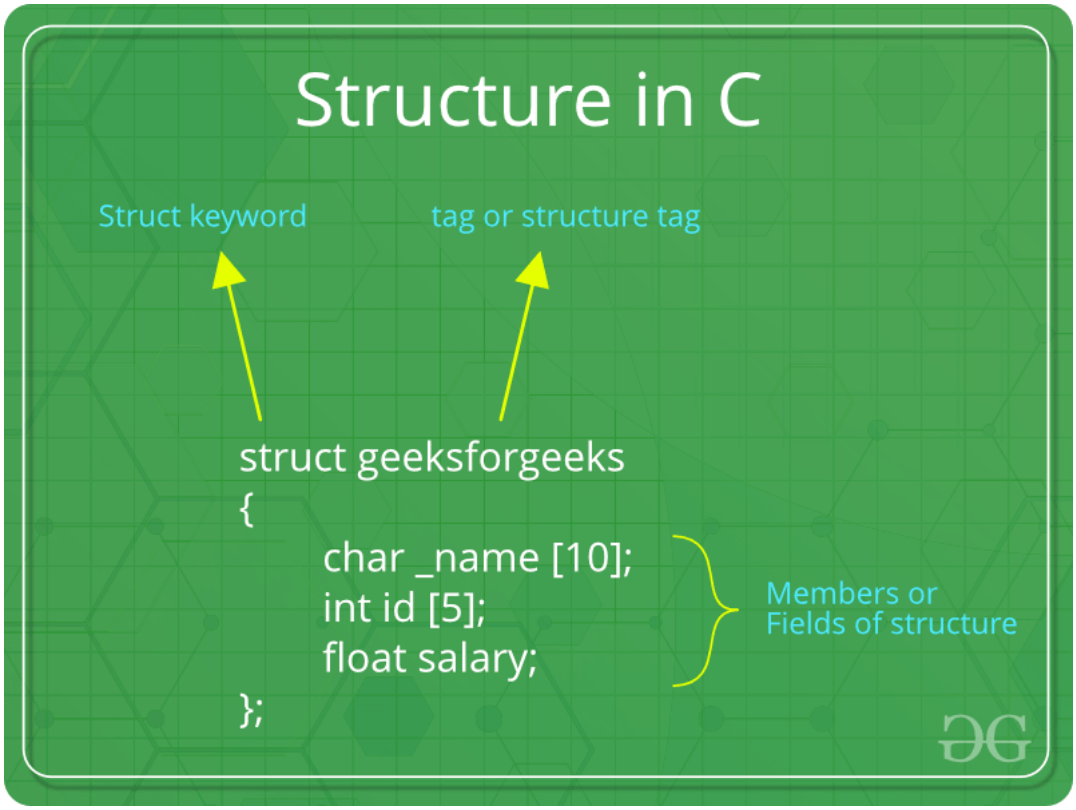
LIVE BATCHES

Output:

```
True
False
```

4. a) Define a structure? Write a program in C to create a database of fifty students to store personal details such as roll no., name and marks. Print all the details of students whose name is entered by the user.

A structure (https://www.geeksforgeeks.org/structures-c/) is a user-defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.



How to create a structure? 'struct' keyword is used to create a structure. Following is an example.

```
struct address {
    char name[50];
    char street[100];
    char city[50];
    char state[20];
    int pin;
};
```

How to declare structure variables? A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

```
// A variable declaration with structure declaration.
struct Point {
    int x, y;
} p1; // The variable p1 is declared with 'Point'

// A variable declaration like basic data types
```

```
struct Point {
    int x, y;
};

int main()
{
    struct Point p1; // The variable p1 is declared like a normal variable
}
```

Note: In C++, the struct keyword is optional before in declaration of a variable. In C, it is mandatory.

Program:

```
#include <stdio.h>
#include <string.h>

struct Student {
    int roll_no;
    char name[100];
    float marks;
};

int main()
{
    int i = 0;
    char n[100];
    struct Student student[50];

    for (i = 0; i < 50; i++) {
        printf("\nEnter details for Student %d", i + 1);

        printf("\nRoll Number: ");
        scanf("%d", &student[i].roll_no);

        printf("\nName: ");
        scanf("%s", student[i].name);

        printf("\nMarks: ");
        scanf("%f", &student[i].marks);
    }

    printf("\nEnter the name of the student whose details you need: ");
```

Run

4. b) What do you mean by macro? Explain types of macro with its examples.

Macros: Macros are piece of code in a program which is given some name. Whenever this name is encountered by the compiler the compiler replaces the name with the actual piece of code. The '#define' directive is used to define a macro. Let us now understand macro definition with the help of a program:

```
#include <iostream>

// macro definition
#define LIMIT 5
int main()
{
    for (int i = 0; i < LIMIT; i++) {
        std::cout << i << "\n";
    }

    return 0;
}
```

Run

Output:

```
0
1
2
3
4
```

Output:

```
0
1
2
3
4
```

In the above program, when the compiler executes the word LIMIT it replaces it with 5. The word 'LIMIT' in macro definition is called macro template and '5' is macro expansion.
Note: There is no semi-colon(';') at the end of macro definition. Macro definitions do not need a semi-colon to end.

Macros with arguments: We can also pass arguments to macros. Macros defined with arguments works similarly as functions. Let us understand this with a program:

```
1
2  #include <iostream>
3
4  // macro with parameter
5  #define AREA(l, b) (l * b)
6  int main()
7  {
8      int l1 = 10, l2 = 5, area;
9
10     area = AREA(l1, l2);
11
12     std::cout << "Area of rectangle is: " << area;
13
14     return 0;
15 }
16
```

Run

Output:

```
Area of rectangle is: 50
```

Output:

```
Area of rectangle is: 50
```

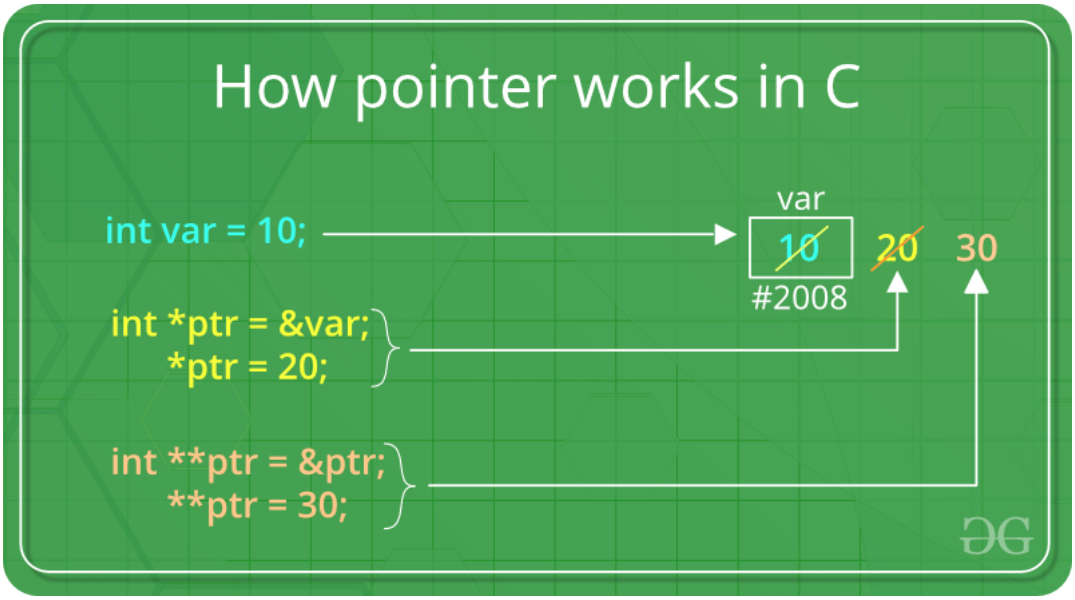
We can see from the above program that whenever the compiler finds AREA(l, b) in the program it replaces it with the statement (l*b) . Not only this, the values passed to the macro template AREA(l, b) will also be replaced in the statement (l*b). Therefore AREA(10, 5) will be equal to 10*5.

5. a)What do you mean by pointers (https://www.geeksforgeeks.org/pointers-c-examples/)? How pointer variables are initialized? Write a program to sort given numbers using pointers.

Pointers are symbolic representation of addresses. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures. It's general declaration in C/C++ has the format:

Syntax:

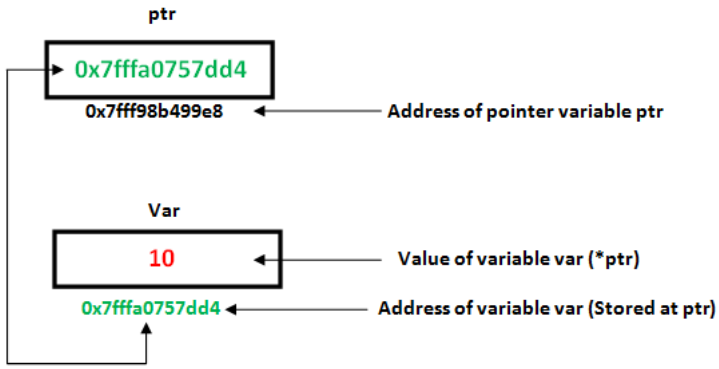
```
datatype *var_name;

int *ptr;   //ptr can point to an address which holds int data
```



How to use a pointer?

- Define a pointer variable
- Assigning the address of a variable to a pointer using unary operator (&) which returns the address of that variable.
- Accessing the value stored in the address using unary operator (*) which returns the value of the variable located at the address specified by its operand.

The reason we associate data type to a pointer is **that it knows how many bytes the data is stored in**. When we increment a pointer, we increase the pointer by the size of data type to which it points.



(https://media.geeksforgeeks.org/wp-content/uploads/pointers-in-c.png)

5. b) A five digit positive integer is entered through the keyboard. Write a C function to calculate sum of digits of the 5 digit number (https://www.geeksforgeeks.org/program-for-sum-the-digits-of-a-given-number/)

i. **Without using Recursion (https://www.geeksforgeeks.org/program-for-sum-the-digits-of-a-given-number/):**

```
// C program to compute sum of digits in
// number.
#include <stdio.h>

/* Function to get sum of digits */
int getSum(int n)
{
    int sum = 0;
    while (n != 0) {
        sum = sum + n % 10;
        n = n / 10;
    }
    return sum;
}

int main()
{
    int n = 687;
    printf(" %d ", getSum(n));
    return 0;
}
```

Run

Output:

```
21
```


ii. Using Recursion (https://www.geeksforgeeks.org/sum-digit-number-using-recursion/):

```c
// Recursive C program to find sum of digits
// of a number
#include <stdio.h>

// Function to check sum of digit using recursion
int sum_of_digit(int n)
{
    if (n == 0)
        return 0;
    return (n % 10 + sum_of_digit(n / 10));
}

// Driven Program to check above
int main()
{
    int num = 12345;
    int result = sum_of_digit(num);
    printf("Sum of digits in %d is %d\n", num, result);
    return 0;
}

```

Run

Output:

```
Sum of digits in 12345 is 15
```

- AKTU 1st Year Sem 2 Solved Paper 2016-17 | COMP. SYSTEM & C PROGRAMMING | Sec A

Paper download link: Paper | Sem 2 | 2016-17 (https://media.geeksforgeeks.org/wp-content/uploads/Paper-Sem-2-2016-17.pdf)

B.Tech.
(SEM-II) THEORY EXAMINATION 2016-17
COMPUTER SYSTEM & PROGRAMMING IN C

Time: 3hrs
Total Marks: 100

Note:-

- There are **three** sections. Section **A** carries **20** marks, Section **B** carries **30** marks and Section **C** carries **50** marks.
- Attempt all questions. Marks are indicated against each question.
- Assume suitable data wherever necessary.

Section – A

1. Explain the following: (2*10 = 20)

a. **What is meant by modular programming approach (https://www.geeksforgeeks.org/modular-approach-in-programming/)?**
Modular programming is the process of subdividing a computer program into separate sub-programs. A module is a separate software component. It can often be used in a variety of applications and functions with other components of the system.
   - Some programs might have thousands or millions of lines and to manage such programs it becomes quite difficult as there might be too many of syntax errors or logical errors present in the program, so to manage such type of programs concept of **modular programming** approached.
   - Each sub-module contains something necessary to execute only one aspect of the desired functionality.
   - Modular programming emphasis on breaking of large programs into small problems to increase the maintainability, readability of the code and to make the program handy to make any changes in future or to correct the errors.

b. **What is operator (https://www.geeksforgeeks.org/operators-c-c/)?**
We can define operators as symbols that helps us to perform specific mathematical and logical computations on operands. In other words we can say that an operator operates the operands.
For example, consider the below statement:

```
c = a + b;
```

Here, '+' is the operator known as *addition operator* and 'a' and 'b' are operands. The addition operator tells the compiler to add both of the operands 'a' and 'b'.

c. **What is structured programming approach (https://www.geeksforgeeks.org/structured-programming-approach-with-advantages-and-disadvantages/)?**
Structured Programming Approach, as the word suggests, can be defined as a programming approach in which the program is made as a single structure. It means that the code will execute the instruction by instruction one after the other. It doesn't support the possibility of jumping from one instruction to some other with help of any statement like GOTO, etc. Therefore, the instructions in this approach will be executed in a serial and structured manner. The languages that support Structured programming approach are:
   - C
   - C++
   - Java
   - C#

   ..etc

d. **What is type conversion (https://www.geeksforgeeks.org/type-conversion-c/)?**
When you assign the value of one data type to another, the two types might not be compatible with each other. If the data types are compatible, then the compiler will perform the conversion known as Type Conversion.

There are two types of Type Conversion:
   1. Implicit Type Conversion
   2. Explicit Type Converstion

e. **Write a function to interchange the two values of two variables without using third variable (https://www.geeksforgeeks.org/swap-two-numbers-without-using-temporary-variable/).**

```
1
2  void swap(int* xp, int* yp)
3  {
4
5      // Code to swap 'xp' and 'xy'
6      *xp = *xp ^ *yp;
7      *yp = *xp ^ *yp;
8      *xp = *xp ^ *yp;
9  }
10
```

Run

f. **Define function declaration (https://www.geeksforgeeks.org/functions-in-c/).**
Function declaration tells compiler about number of parameters function takes, data-types of parameters and return type of function. Putting parameter names in function declaration is optional in function declaration, but it is necessary to put them in definition.

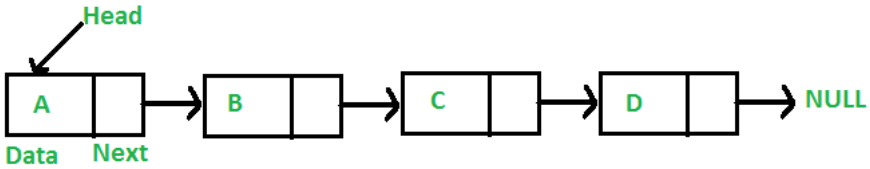
g. **Enlist different file opening modes in C (https://www.geeksforgeeks.org/basics-file-handling-c/).**
File opening modes in C:
   - **“r” -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.

   - **“w” -** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

   - **"a" -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

- **"r+" -** Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.
- **"w+" -** Searches file. If the file exists, its contents are overwritten. If the file doesn't exist a new file is created. Returns NULL, if unable to open file.
- **"a+" -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn't exist, a new file is created. Returns NULL, if unable to open file.

h. **What is meant by linkedlist (https://www.geeksforgeeks.org/linked-list-set-1-introduction/)?**
A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

i. **Write the use of 'switch' statement (https://www.geeksforgeeks.org/switch-statement-cc/)?**
Switch case statements are a substitute for long if statements that compare a variable to several integral values:
- The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.
- Switch is a control statement that allows a value to change control of execution.

j. **Explain various data types (https://www.geeksforgeeks.org/data-types-in-c/)?**
Following are the examples of some very common data types used in C:

- **char:** The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.
- **int:** As the name suggests, an int variable is used to store an integer.
- **float:** It is used to store decimal numbers (numbers with floating point value) with single precision.
- **double:** It is used to store decimal numbers (numbers with floating point value) with double precision.

– AKTU 1st Year Sem 2 Solved Paper 2016-17 | COMP. SYSTEM & C PROGRAMMING | Sec B

Paper download link: Paper | Sem 2 | 2016-17 (https://media.geeksforgeeks.org/wp-content/uploads/Paper-Sem-2-2016-17.pdf)

B.Tech.
(SEM-II) THEORY EXAMINATION 2016-17
COMPUTER SYSTEM & PROGRAMMING IN C

Time: 3hrs
Total Marks: 100

Note:-
- There are **three** sections. Section **A** carries **20** marks, Section **B** carries **30** marks and Section **C** carries **50** marks.
- Attempt all questions. Marks are indicated against each question.
- Assume suitable data wherever necessary.

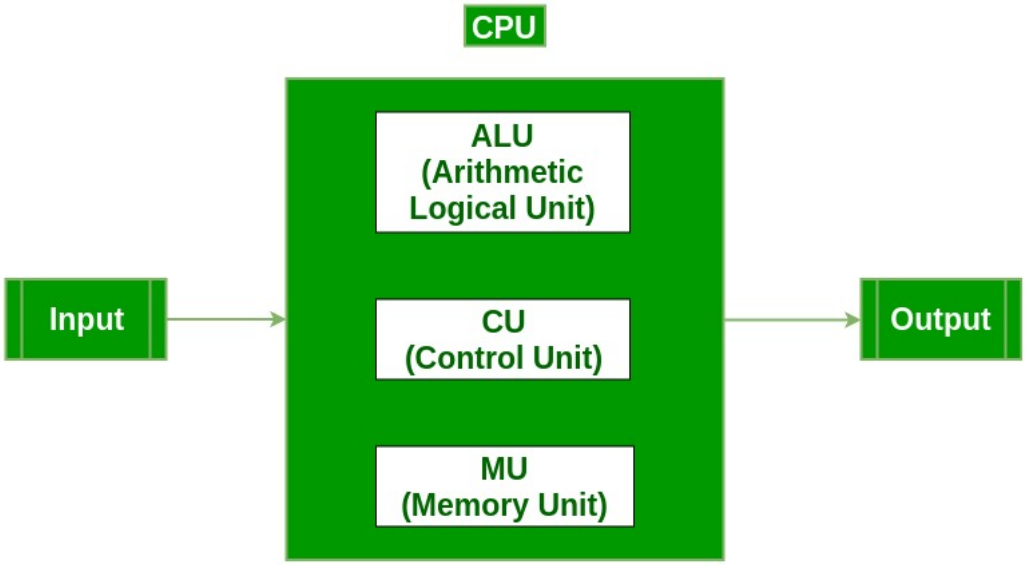
Section – B

2. Attempt any five of the following questions: (5*10 = 50)

a. What is digital computer (https://www.geeksforgeeks.org/functional-components-of-a-computer/)? Also explain block diagram of digital computer in detail.

Digital Computer: A digital computer can be defined as a programmable machine which reads the binary data passed as instructions, processes this binary data, and displays a calculated digital output. Therefore, Digital computers are those that work on the digital data.

Details of Functional Components of a Digital Computer



- **Input Unit :**The input unit consists of input devices that are attached to the computer. These devices take input and convert it into binary language that the computer understands. Some of the common input devices are keyboard, mouse, joystick, scanner etc.
- **Central Processing Unit (CPU0 :** Once the information is entered into the computer by the input device, the processor processes it. The CPU is called the brain of the computer because it is the control center of the computer. It first fetches instructions from memory and then interprets them so as to know what is to be done. If required, data is fetched from memory or input device. Thereafter CPU executes or performs the required computation and then either stores the output or displays on the output device. The CPU has three main components which are responsible for different functions – Arithmetic Logic Unit (ALU), Control Unit (CU) and Memory registers
- **Arithmetic and Logic Unit (ALU) :** The ALU, as its name suggests performs mathematical calculations and takes logical decisions. Arithmetic calculations include addition, subtraction, multiplication and division. Logical decisions involve comparison of two data items to see which one is larger or smaller or equal.
- **Control Unit :** The Control unit coordinates and controls the data flow in and out of CPU and also controls all the operations of ALU, memory registers and also input/output units. It is also responsible for carrying out all the instructions stored in the program. It decodes the fetched instruction, interprets it and sends control signals to input/output devices until the required operation is done properly by ALU and memory.
- **Memory Registers :** A register is a temporary unit of memory in the CPU. These are used to store the data which is directly used by the processor. Registers can be of different sizes(16 bit, 32 bit, 64 bit and so on) and each register inside the CPU has a specific function like storing data, storing an instruction, storing address of a location in memory etc. The user registers can be used by an assembly language programmer for storing operands, intermediate results etc. Accumulator (ACC) is the main register in the ALU and contains one of the operands of an operation to be performed in the ALU.
- **Memory (https://www.geeksforgeeks.org/types-computer-memory-ram-rom/) :** Memory attached to the CPU is used for storage of data and instructions and is called internal memory The internal memory is divided into many storage locations, each of which can store data or instructions. Each memory location is of the same size and has an address. With the help of the address, the computer can read any memory location easily without having to search the entire memory. when a program is executed, it's data is copied to the internal memory ans is stored in the memory till the end of the execution. The internal memory is also called the Primary memory or Main memory. This memory is also called as RAM, i.e. Random Access Memory. The time of access of data is independent of its location in memory, therefore this memory is also called Random Access memory (RAM). Read this for different types of RAMs (https://www.geeksforgeeks.org/different-types-ram-random-access-memory/)
- **Output Unit :** The output unit consists of output devices that are attached with the computer. It converts the binary data coming from CPU to human understandable form. The common output devices are monitor, printer, plotter etc.

b. Write a program that calculate sum of the digits of an integer (https://www.geeksforgeeks.org/program-for-sum-the-digits-of-a-given-number/). For example, the sum of the digit of the number 2155 is 2+1+5+5 or 13. The program should accept any arbitrary number typed by user.

```
// C program to compute sum of digits in
// number.

#include <stdio.h>

/* Function to get sum of digits */
int getSum(int n)
{
    int sum = 0;
    while (n != 0) {
        sum = sum + n % 10;
        n = n / 10;
    }
    return sum;
}

// Driver code
int main()
{

    int n;

    // Get the number
    scanf("%d", &n);
    printf("Enter the number: %d", n);

    // Print the digits of the number
    printf("\nSum of Digits: %d ",
           getSum(n));
```

Run

Output:

```
Enter the number: 32764

Sum of Digits: 22
```

c. Write a program to copy the contents of one array into another in the reverse order.

```
#include <stdio.h>

// Driver code
int main()
{
    int original_arr[10], copied_arr[10], i;

    // Get the numbers in the array
    printf("\nEnter the Elements: ");
    for (i = 0; i < 10; i++) {
        scanf("%d", &original_arr[i]);
        printf("%d, ", original_arr[i]);
    }

    // Copy the elements of the array
    // in the copied_arr in Reverse Order
    for (i = 0; i < 10; i++) {
        copied_arr[i] = original_arr[10 - i - 1];
    }

    // Print the original_arr
    printf("\nOriginal array: ");
    for (i = 0; i < 10; i++) {
        printf("%d ", original_arr[i]);
    }

    // Print the copied array
    printf("\nResultant array: ");
    for (i = 0; i < 10; i++) {
```

Run

Output:

```
Enter the Elements: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
Original array: 1 2 3 4 5 6 7 8 9 10
Resultant array: 10 9 8 7 6 5 4 3 2 1
```

d. What is Operating System (https://www.geeksforgeeks.org/operating-systems/)? Also define types and functions of OS (https://www.geeksforgeeks.org/operating-system-types-operating-systems-awaiting-author/).

Operating System - Definition:

- An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
- A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.
- An operating system is concerned with the allocation of resources and services, such as memory, processors, devices and information. The operating system correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

Functions of Operating system - Operating system performs three functions:

1. Convenience: An OS makes a computer more convenient to use.
2. Efficiency: An OS allows the computer system resources to be used in an efficient manner.
3. Ability to Evolve: An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions without at the same time interfering with service.

Types of Operating System (https://www.geeksforgeeks.org/operating-system-types-operating-systems-awaiting-author/) -

- Batch Operating System- Sequence of jobs in a program on a computer without manual interventions.
- Time sharing operating System- allows many users to share the computer resources.(Max utilization of the resources).
- Distributed operating System- Manages a group of different computers and make appear to be a single computer.
- Network operating system- computers running in different operating system can participate in common network (It is used for security purpose).
- Real time operating system – meant applications to fix the deadlines.

Examples of Operating System are -

- Windows (GUI based, PC)
- GNU/Linux (Personal, Workstations, ISP, File and print server, Three-tier client/Server)
- macOS (Macintosh), used for Apple's personal computers and work stations (MacBook, iMac).
- Android (Google's Operating System for smartphones/tablets/smartwatches)
- iOS (Apple's OS for iPhone, iPad and iPod Touch)

e. Write a program to multiply two matrices (read size and number of element of matrices from the keyboard).

```c

// C program to multiply two square matrices.

#include <stdio.h>

const int MAX = 100;

// Function to print Matrix
void printMatrix(int M[][MAX], int rowSize, int colSize)
{
    for (int i = 0; i < rowSize; i++) {
        for (int j = 0; j < colSize; j++)
            printf("%d ", M[i][j]);

        printf("\n");
    }
}

// Function to multiply two matrices A[][] and B[][]
void multiplyMatrix(int row1, int col1, int A[][MAX],
                    int row2, int col2, int B[][MAX])
{
    int i, j, k;

    // Matrix to store the result
    int C[MAX][MAX];

    // Check if multiplication is Possible
    if (row2 != col1) {
        printf("Not Possible\n");
```

Run

Output:

```
Enter the number of rows of First Matrix: 2
Enter the number of columns of First Matrix: 3
Enter the elements of First Matrix:
A[0][0]: 1
A[0][1]: 2
A[0][2]: 3
A[1][0]: 4
A[1][1]: 5
A[1][2]: 6

Enter the number of rows of Second Matrix: 3
Enter the number of columns of Second Matrix: 2
Enter the elements of First Matrix:
B[0][0]: 1
B[0][1]: 2
B[1][0]: 3
B[1][1]: 4
B[2][0]: 5
B[2][1]: 6

First Matrix:
1 2 3
4 5 6

Second Matrix:
1 2
3 4
5 6

Resultant Matrix:
22 28
49 64
```


f. Write a program to sort an integer array in ascending order (https://www.geeksforgeeks.org/selection-sort/).

```c

// C program for implementation of selection sort
#include <stdio.h>

void swap(int* xp, int* yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n - 1; i++) {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i + 1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}

/* Function to print an array */
void printArray(int arr[], int size)
```

Run

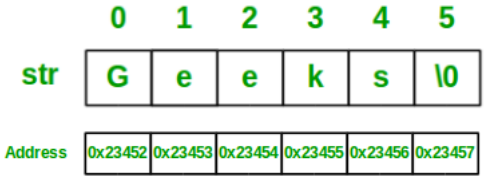
Output:

```
Sorted array:

11 12 22 25 64
```

g. **What is string (https://www.geeksforgeeks.org/strings-in-c-2/)? Also explain different string functions (https://www.geeksforgeeks.org/commonly-used-string-functions-in-c-c-with-examples/). Write a user define function to compare two strings where they are identical or not.**
Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.

Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string in **C programming** language.

```
char str_name[size];
```



Some of the most commonly used String functions are:

- **strcat (https://www.geeksforgeeks.org/strcat-vs-strncat-c/)**: The strcat() function will append a copy of the source string to the end of destination string.
- **strchr (https://www.geeksforgeeks.org/strrchr-function-in-cpp/)**: In C/C++, strrchr() is a predefined function used for string handling. cstring is the header file required for string functions.

  This function Returns a pointer to the last occurrence of a character in a string.
  The character whose last occurrence we want to find in passed as the second argument to the function and the string in which we have to find the character is passed as the first argument to the function.

- **strcmp (https://www.geeksforgeeks.org/strcmp-in-c-cpp/)**: strcmp() is a built-in library function and is declared in **<string.h>** header file. This function takes two strings as arguments and compare these two strings lexicographically.

- **strcpy (https://www.geeksforgeeks.org/strcpy-in-c-cpp/)**: strcpy() is a standard library function in C/C++ and is used to copy one string to another. In C it is present in **string.h** header file and in C++ it is present in **cstring** header file.

- **strlen (https://www.geeksforgeeks.org/strlen-function-in-c/)**: The **strlen()** function calculates the length of a given string.The **strlen()** function is defined in **string.h** header file. It doesn't count null character '\0'.

- **strncat (https://www.geeksforgeeks.org/strncat-function-in-c-cpp/)**: In C/C++, strncat() is a predefined function used for string handling. **string.h** is the header file required for string functions.

  This function appends not more than **n** characters from the string pointed to by src to the end of the string pointed to by dest plus a terminating Null-character. The initial character of string(src) overwrites the Null-character present at the end of string(dest). Thus, length of the string(dest) becomes strlen(dest)+n. But, if the length of the string(src) is less than **n**, only the content up to the terminating null-character is copied and length of the string(dest) becomes strlen(src) + strlen(dest).

  The behavior is undefined if
  - the strings overlap.
  - the dest array is not large enough to append the contents of src.

- **strncmp (https://www.geeksforgeeks.org/stdstrncmp-in-c/)**: **std::strncmp()** function lexicographically compares not more than count characters from the two null-terminated strings and returns an integer based on the outcome.
  - This function takes two strings and a number **num** as arguments and compare at most first **num** bytes of both the strings.
  - **num** should be at most equal to the length of the longest string. If **num** is defined greater than the string length than comparison is done till the null-character('\0') of either string.
  - This function compares the two strings lexicographically. It starts comparison from the first character of each string. If they are equal to each other, it continues and compare the next character of each string and so on.
  - This process of comparison stops until a terminating null-character of either string is reached or **num** characters of both the strings matches.

- **strncpy (https://www.geeksforgeeks.org/why-strcpy-and-strncpy-are-not-safe-to-use/)**: The strncpy() function is similar to strcpy() function, except that at most n bytes of src are copied. If there is no NULL character among the first n character of src,

the string placed in dest will not be NULL-terminated. If the length of src is less than n, strncpy() writes additional NULL character to dest to ensure that a total of n character are written.

- **strrchr (https://www.geeksforgeeks.org/strrchr-function-in-c-c/)**: The **strrchr()** function in C/C++ locates the last occurrence of a character in a string. It returns a pointer to the last occurrence in the string. The terminating null character is considered part of the C string. Therefore, it can also be located to retrieve a pointer to the end of a string. It is defined in **cstring** header file.

Program to check if two strings are identical or not (https://www.geeksforgeeks.org/program-to-check-if-two-strings-are-same-or-not/):

```
// C program to check if
// two strings are identical

#include <stdio.h>
#include <string.h>

int main()
{

    char string1[100], string2[100];

    // Get the strings which
    // is to be checked
    scanf("%s", string1);
    printf("Enter the first string: %s", string1);

    // Get the strings which
    // is to be checked
    scanf("%s", string2);
    printf("\nEnter the second string: %s", string2);

    // Check if both strings are equal
    printf("\nAre both strings same: ");

    if (strcmp(string1, string2) == 0) {
        printf("Yes");
    }
    else {
        printf("No");
```

Run

Output:

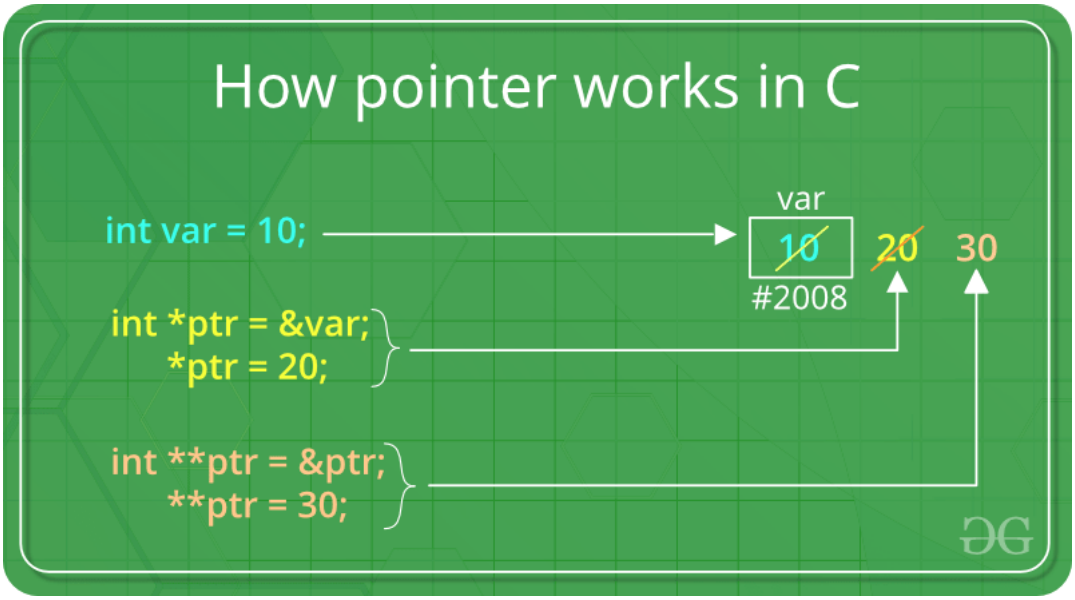
```
Enter the first string: GeeksForGeeks
Enter the second string: GeeksForGeeks
Are both strings same: Yes
```

h. **Define the concept of pointer (https://www.geeksforgeeks.org/pointers-c-examples/)? Also define the dynamic memory allocation and its various functions.**

Pointers are symbolic representation of addresses. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures. It's general declaration in C/C++ has the format:

Syntax:

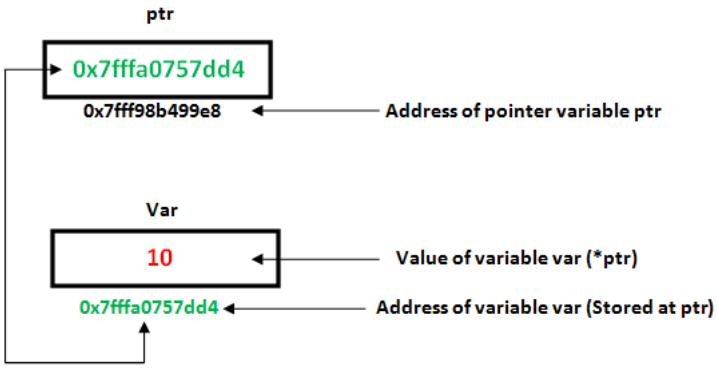
```
datatype *var_name;

int *ptr;   //ptr can point to an address which holds int data
```



How to use a pointer?

- Define a pointer variable
- Assigning the address of a variable to a pointer using unary operator (&) which returns the address of that variable.
- Accessing the value stored in the address using unary operator (*) which returns the value of the variable located at the address specified by its operand.

The reason we associate data type to a pointer is **that it knows how many bytes the data is stored in**. When we increment a pointer, we increase the pointer by the size of data type to which it points.



(https://media.geeksforgeeks.org/wp-content/uploads/pointers-in-c.png)

Dynamic Memory Allocation in C (https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/): It can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime.

C provides some functions to achieve these tasks. There are 4 library functions provided by C defined under **<stdlib.h>** header file to facilitate dynamic memory allocation in C programming. They are:

1. malloc()
2. calloc()
3. free()
4. realloc()

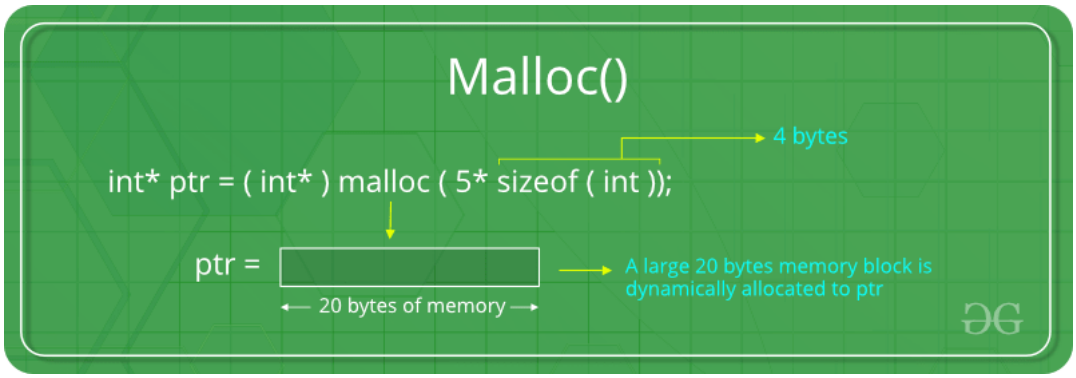
Lets see each of them in detail.

1. malloc()

"malloc" or **"memory allocation"** method is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form.

Syntax:

```
ptr = (cast-type*) malloc(byte-size)

For Example:
ptr = (int*) malloc(100 * sizeof(int));

Since the size of int is 4 bytes,
this statement will allocate 400 bytes of memory.
And, the pointer ptr holds the address
of the first byte in the allocated memory.
```



If the space is insufficient, allocation fails and returns a NULL pointer.

Example:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{

    // This pointer will hold the
    // base address of the block created
    int* ptr;
    int n, i, sum = 0;

    // Get the number of elements for the array
    n = 5;
    printf("Enter number of elements: %d\n", n);

    // Dynamically allocate memory using malloc()
    ptr = (int*)malloc(n * sizeof(int));

    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {

        // Memory has been successfully allocated
        printf("Memory successfully allocated using malloc.\n");

```

Run

Output:

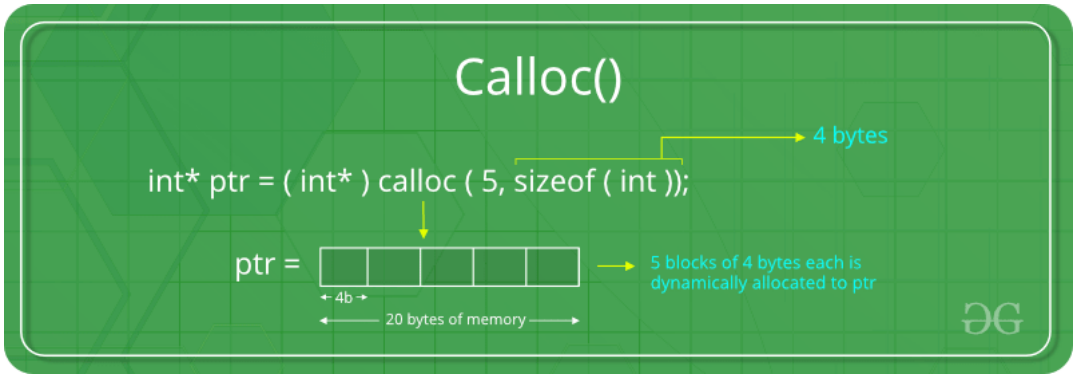
```
Enter number of elements: 5
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5,
```

2. calloc()

"calloc" or **"contiguous allocation"** method is used to dynamically allocate the specified number of blocks of memory of the specified type. It initializes each block with a default value '0'.

Syntax:

```
ptr = (cast-type*)calloc(n, element-size);

For Example:
ptr = (float*) calloc(25, sizeof(float));

This statement allocates contiguous space in memory
for 25 elements each with the size of float.
```



If the space is insufficient, allocation fails and returns a NULL pointer.

Example:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{

    // This pointer will hold the
    // base address of the block created
    int* ptr;
    int n, i, sum = 0;

    // Get the number of elements for the array
    n = 5;
    printf("Enter number of elements: %d\n", n);

    // Dynamically allocate memory using calloc()
    ptr = (int*)calloc(n, sizeof(int));

    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {

        // Memory has been successfully allocated
        printf("Memory successfully allocated using calloc.\n");

```

Run

Output:

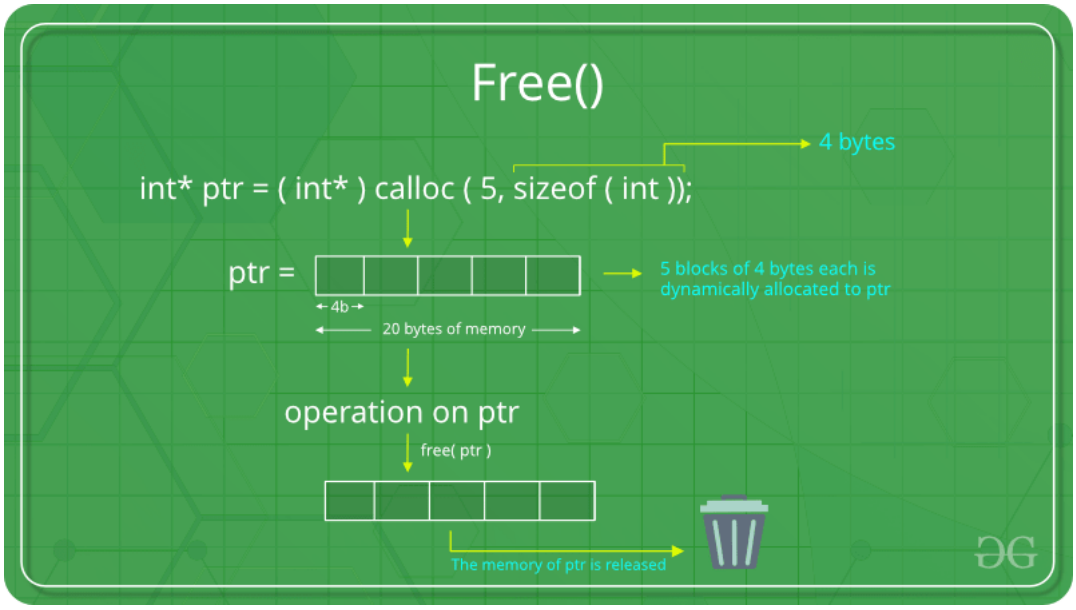
```
Enter number of elements: 5
Memory successfully allocated using calloc.
The elements of the array are: 1, 2, 3, 4, 5,
```

3. free()

"free" method is used to dynamically de-allocate the memory. The memory allocated using functions malloc() and calloc() are not de-allocated on their own. Hence the free() method is used, whenever the dynamic memory allocation takes place. It helps to reduce wastage of memory by freeing it.

Syntax:

```
free(ptr);
```



Example:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{

    // This pointer will hold the
    // base address of the block created
    int *ptr, *ptr1;
    int n, i, sum = 0;

    // Get the number of elements for the array
    n = 5;
    printf("Enter number of elements: %d\n", n);

    // Dynamically allocate memory using malloc()
    ptr = (int*)malloc(n * sizeof(int));

    // Dynamically allocate memory using calloc()
    ptr1 = (int*)calloc(n, sizeof(int));

    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL || ptr1 == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {

```

Run

Output:

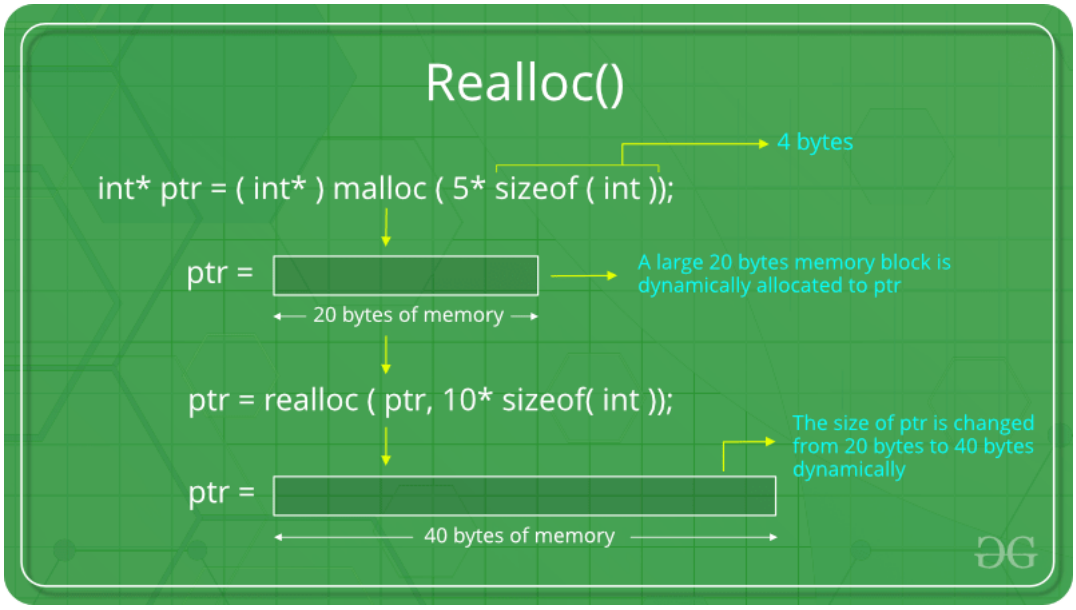
```
Enter number of elements: 5
Memory successfully allocated using malloc.
Malloc Memory successfully freed.

Memory successfully allocated using calloc.
Calloc Memory successfully freed.
```


4. realloc()

"realloc" or **"re-allocation"** method is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to **dynamically re-allocate memory**.

Syntax:

```
ptr = realloc(ptr, newSize);

where ptr is reallocated with new size 'newSize'.
```



If the space is insufficient, allocation fails and returns a NULL pointer.

Example:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{

    // This pointer will hold the
    // base address of the block created
    int* ptr;
    int n, i, sum = 0;

    // Get the number of elements for the array
    n = 5;
    printf("Enter number of elements: %d\n", n);

    // Dynamically allocate memory using calloc()
    ptr = (int*)calloc(n, sizeof(int));

    // Check if the memory has been successfully
    // allocated by malloc or not
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
```

```
    }
    else {

        // Memory has been successfully allocated
        printf("Memory successfully allocated using calloc.\n");

```

Run

Output:

```
Enter number of elements: 5
Memory successfully allocated using calloc.
The elements of the array are: 1, 2, 3, 4, 5,

Enter the new size of the array: 10
Memory successfully re-allocated using realloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

- AKTU 1st Year Sem 2 Solved Paper 2016-17 | COMP. SYSTEM & C PROGRAMMING | Sec C

Paper download link: Paper | Sem 2 | 2016-17 (https://media.geeksforgeeks.org/wp-content/uploads/Paper-Sem-2-2016-17.pdf)

B.Tech.
(SEM-II) THEORY EXAMINATION 2016-17
COMPUTER SYSTEM & PROGRAMMING IN C

Time: 3hrs
Total Marks: 100

Note:-

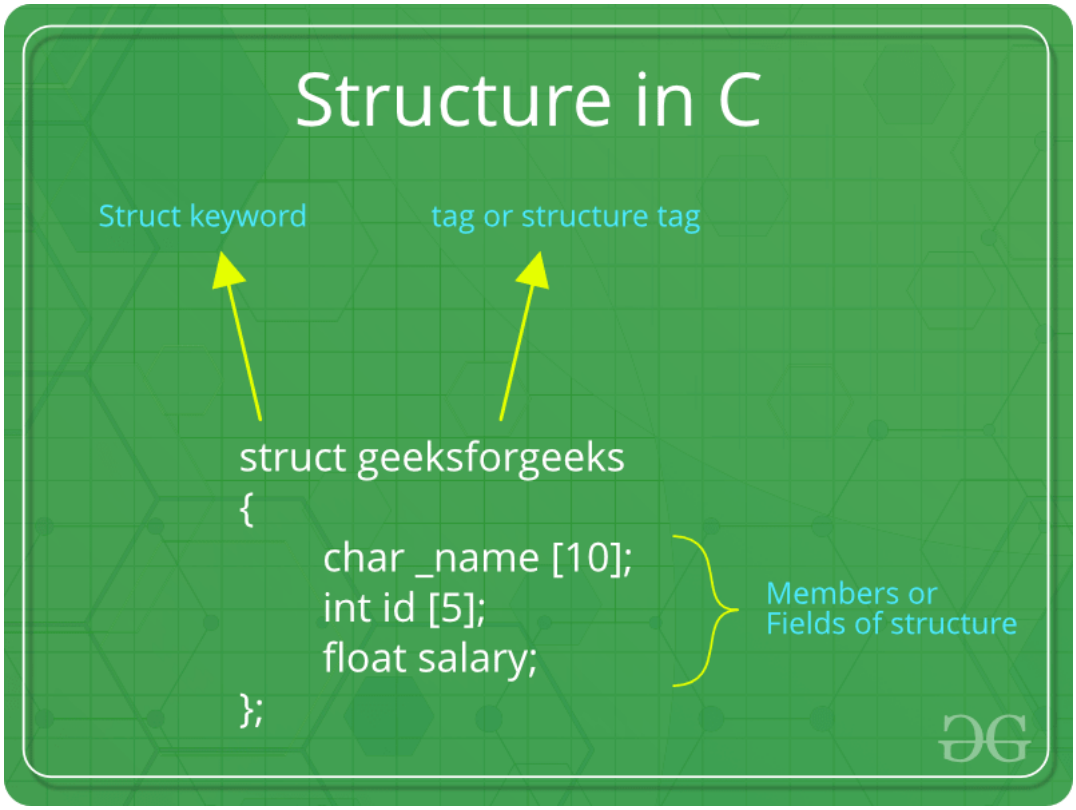
- There are **three** sections. Section **A** carries **20** marks, Section **B** carries **30** marks and Section **C** carries **50** marks.
- Attempt all questions. Marks are indicated against each question.
- Assume suitable data wherever necessary.

Section – C

Attempt any two of the following questions: (2*15 = 30)

3) Define structure with syntax. Also, write a program that compares two given dates. To store date use structure say date that contains three members namely date, month and year. If the dates are equal then display message as "Equal" otherwise "Unequal".

- A structure (https://www.geeksforgeeks.org/structures-c/) is a user-defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.



How to create a structure? 'struct' keyword is used to create a structure. Following is an example.

```
struct address {
    char name[50];
    char street[100];
    char city[50];
    char state[20];
    int pin;
};
```

How to declare structure variables? A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

```
// A variable declaration with structure declaration.
struct Point {
    int x, y;
} p1; // The variable p1 is declared with 'Point'

// A variable declaration like basic data types
struct Point {
    int x, y;
};

int main()
{
    struct Point p1; // The variable p1 is declared like a normal variable
}
```

Note: In C++, the struct keyword is optional before in declaration of a variable. In C, it is mandatory.

Program that compares two given dates:

```
#include <stdio.h>

// Declaring the structure of Date
struct Date {
    int date;
    int month;
    int year;
};

// Driver code
int main()
{
    int date1, date2, month1,
        month2, year1, year2;

    // Get the first date
```

```
    scanf("%d", &date1);
    printf("Enter the first date: %d", date1);
    scanf("%d", &month1);
    printf("\nEnter the first month: %d", month1);
    scanf("%d", &year1);
    printf("\nEnter the first year: %d", year1);

    // Initialise the structure with first date
    struct Date Date1 = { date1, month1, year1 };

    // Get the second date
    scanf("%d", &date2);
    printf("\nEnter the second date: %d", date2);
```


Run

Output:

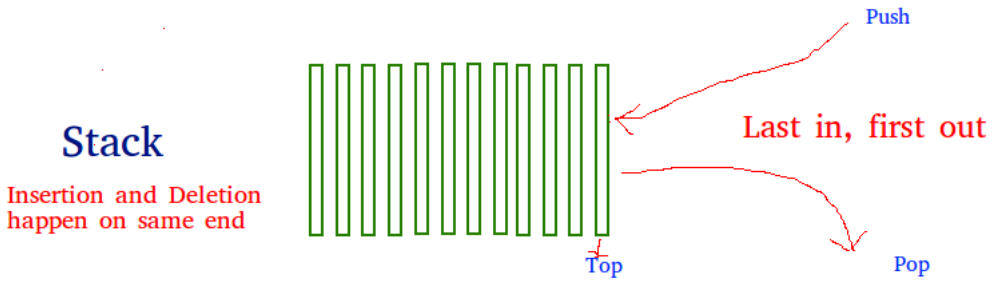
```
Enter the first date: 10
Enter the first month: 11
Enter the first year: 2018
Enter the second date: 10
Enter the second month: 11
Enter the second year: 2018
The given dates are: Equal
```

4 Write short note on following (Any two):

- (i) Stack with push and pop operation (https://www.geeksforgeeks.org/stack-data-structure-introduction-program/): Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

Mainly the following three basic operations are performed in the stack:

- Push: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- Pop: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- Peek or Top: Returns top element of stack.
- isEmpty: Returns true if stack is empty, else false.



(https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2013/03/stack.png)

How to understand a stack practically? There are many real life examples of stack. Consider the simple example of plates stacked over one another in canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO/FILO order.

Time Complexities of operations on stack:

push(), pop(), isEmpty() and peek() all take O(1) time. We do not run any loop in any of these operations.

Applications of stack:

- Balancing of symbols (https://www.geeksforgeeks.org/check-for-balanced-parentheses-in-an-expression/)
- Infix to Postfix (http://quiz.geeksforgeeks.org/stack-set-2-infix-to-postfix/) /Prefix conversion
- Redo-undo features at many places like editors, photoshop.
- Forward and backward feature in web browsers
- Used in many algorithms like Tower of Hanoi, (https://www.geeksforgeeks.org/recursive-functions/) tree traversals (https://www.geeksforgeeks.org/618/), stock span problem (https://www.geeksforgeeks.org/the-stock-span-problem/), histogram problem (https://www.geeksforgeeks.org/largest-rectangular-area-in-a-histogram-set-1/).

- Other applications can be Backtracking, Knight tour problem (https://www.geeksforgeeks.org/backtracking-set-1-the-knights-tour-problem/), rat in a maze (https://www.geeksforgeeks.org/backttracking-set-2-rat-in-a-maze/), N queen problem (https://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/) and sudoku solver (https://www.geeksforgeeks.org/backtracking-set-7-suduku/)
- In Graph Algorithms like Topological Sorting (https://www.geeksforgeeks.org/topological-sorting/) and Strongly Connected Components (https://www.geeksforgeeks.org/strongly-connected-components/)

Implementation: There are two ways to implement a stack:

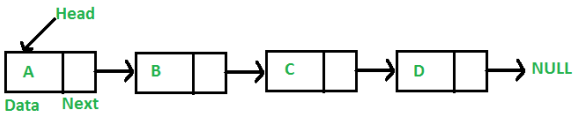
- Using array
- Using linked list

Implementing Stack using Arrays

```
// C program for array implementation of stack
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

// A structure to represent a stack
struct Stack {
    int top;
    unsigned capacity;
    int* array;
};

// function to create a stack of given capacity. It initializes size of
// stack as 0
struct Stack* createStack(unsigned capacity)
{
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
    stack->array = (int*)malloc(stack->capacity * sizeof(int));
    return stack;
}

// Stack is full when top is equal to the last index
int isFull(struct Stack* stack)
{
    return stack->top == stack->capacity - 1;
}

```

Run

- **(ii) Linked List (http://www.geeksforgeeks.org/data-structures/linked-list/):** Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.



(https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2013/03/Linkedlist.png)
Why Linked List? Arrays can be used to store linear data of similar types, but arrays have following limitations.
1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.
2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.

For example, in a system if we maintain a sorted list of IDs in an array id[].

id[] = [1000, 1010, 1050, 2000, 2040].

And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000). Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

Advantages over arrays 1) Dynamic size

2) Ease of insertion/deletion

Drawbacks: 1) Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists efficiently with its default implementation. Read about it here (https://www.geeksforgeeks.org/binary-search-on-singly-linked-list/).
2) Extra memory space for a pointer is required with each element of the list.
3) Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

Representation: A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.
Each node in a list consists of at least two parts:
1) data
2) Pointer (Or Reference) to the next node
In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.

```
1
2  // A linked list node
3  struct Node {
4      int data;
5      struct Node* next;
6  };
7
```

Run

- **(iii) Command line argument (https://www.geeksforgeeks.org/command-line-arguments-in-c-cpp/):** The most important function of C/C++ is main() function. It is mostly defined with a return type of int and without parameters :

```
int main() { /* ... */ }
```

We can also give command-line arguments in C and C++. Command-line arguments are given after the name of the program in command-line shell of Operating Systems.
To pass command line arguments, we typically define main() with two arguments : first argument is the number of command line arguments and second is list of command-line arguments.

```
int main(int argc, char *argv[]) { /* ... */ }
```

or

```
int main(int argc, char **argv) { /* ... */ }
```

- **argc (ARGument Count)** is int and stores number of command-line arguments passed by the user including the name of the program. So if we pass a value to a program, value of argc would be 2 (one for argument and one for program name)
- The value of argc should be non negative.
- **argv(ARGument Vector)** is array of character pointers listing all the arguments.
- If argc is greater than zero, the array elements from argv[0] to argv[argc-1] will contain pointers to strings.
- Argv[0] is the name of the program, After that till argv[argc-1] every element is command -line arguments.

5. What are the different file opening modes in C (https://www.geeksforgeeks.org/basics-file-handling-c/). Suppose a file contains student's records with each record containing name and age of a student. Write a C program to read these records and display them in sorted order by name.

- File opening modes in C:
  - **"r" -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.
  - **"w" -** Searches file. If the file exists, its contents are overwritten. If the file doesn't exist, a new file is created. Returns NULL, if unable to open file.
  - **"a" -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn't exist, a new file is created. Returns NULL, if unable to open file.
  - **"r+" -** Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.

- **"w+" -** Searches file. If the file exists, its contents are overwritten. If the file doesn't exist a new file is created. Returns NULL, if unable to open file.

- **"a+" -** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn't exist, a new file is created. Returns NULL, if unable to open file.

C program to read these records and display them in sorted order by name:

```
// C program to read Student records
// like id, name and age,
// and display them in sorted order by Name

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// struct person with 3 fields
struct Student {
    char* name;
    int id;
    char age;
};

// setting up rules for comparison
// to sort the students based on names
int comparator(const void* p, const void* q)
{
    return strcmp(((struct Student*)p)->name,
                  ((struct Student*)q)->name);
}

// Driver program
int main()
{
    int i = 0, n = 5;

    struct Student arr[n];
```

Run

Output:

```
Unsorted Student Records:
Id = 1, Name = bd, Age = 12
Id = 2, Name = ba, Age = 10
Id = 3, Name = bc, Age = 8
Id = 4, Name = aaz, Age = 9
Id = 5, Name = az, Age = 10


Student Records sorted by Name:
Id = 4, Name = aaz, Age = 9
Id = 5, Name = az, Age = 10
Id = 2, Name = ba, Age = 10
Id = 3, Name = bc, Age = 8
Id = 1, Name = bd, Age = 12
```

(https://www.facebook.com/geeksforgeeks.org/)(https://www.instagram.com/geeks_for_geeks/)(https://in.linkedin.com/company/geeksforgeek

Company

About Us (https://www.geeksforgeeks.org/about/)

Careers (https://www.geeksforgeeks.org/careers/)

Privacy Policy (https://www.geeksforgeeks.org/privacy-policy/)

Contact Us (https://www.geeksforgeeks.org/about/contact-us/)

Learn

Algorithms (https://www.geeksforgeeks.org/fundamentals-of-algorithms/)

Data Structures (https://www.geeksforgeeks.org/data-structures/)

Languages (https://www.geeksforgeeks.org/category/program-output/)

CS Subjects (https://www.geeksforgeeks.org/articles-on-computer-science-subjects-gq/)

Video Tutorials (https://www.youtube.com/geeksforgeeksvideos/)

Practice

Courses (https://practice.geeksforgeeks.org/courses/)

Company-wise (https://practice.geeksforgeeks.org/company-tags/)

Topic-wise (https://practice.geeksforgeeks.org/topic-tags/)

How to begin? (https://practice.geeksforgeeks.org/faq.php)

Contribute

Write an Article (https://www.geeksforgeeks.org/contribute/)

Write Interview Experience (https://www.geeksforgeeks.org/write-interview-experience/)

Internships (https://www.geeksforgeeks.org/internship/)

Videos (https://www.geeksforgeeks.org/how-to-contribute-videos-to-geeksforgeeks/)

(https://in.linkedin.com/company/geeksforgeeks)