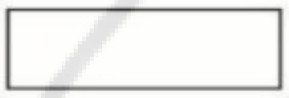
UNIT 1: Basic of C-Programming

1. Define Algorithm ? Pseudo Code.

Ans: Algorithm Is Step By Step Procedure For Solving A Problem In A Finite Amount Of Time With A Finite Amount Of Data.

2. Symbols of Flowchart

Ans:

Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

3. All the Algorithms / Program / Flowchart thought in lecture

1. Write an algorithm to add two numbers entered by user.

Ans:

Step 1: start

Step2: declare variable n1,n2 and sum.

Step3: read values n1,n2.

Step4: add n1 and n2 and assign the result to sum.

Sum ⟵ n1+n2.

Step5: display sum.

Step6: stop.

2. Largest among 3 numbers.

Ans:

Step1: Start

Step 2: Declare variables a, b and c.

Step 3: Read variables a, b and c.

Step 4: If a>b If a>c Display a is the largest number Else Display c is the largest number Else If b>c Display b is the largest number. Else Display c is the greatest number.

Step 5: Stop

3. Factorial of a number.

Ans:

Step1 :Start

Step 2: Declare variables n,factorial and i

Step3: Initialize variables factorial⟵1,i⟵1

Step 4: Read value of n

Step 5: Repeat the steps until i=n 5.1 factorial-factorial*1 5.2.1-1+1

Step 6: Display factorial

Step 7: Stop

4. Prime number or not

Ans:

Step 1: Start
Step 2: Declare variables n,i,flag.
Step 3: Initialize variables
flag⟵1
i ⟵ 2
Step 4: Read n from user.
Step 5: Repeat the steps until i<(n/2)
5.1 If remainder of n÷i equals 0
flag⟵0
Go to step6
5.2 i-i+1
Step 6: If flag=0
Display n is not prime else

Display n is prime
Step 7: Stop

5. Even or Odd

Ans:

Step 1- Start
Step 2-Read/input the number.
Step 3- if n%2==0 then number is even.
Step 4- else number is odd.
Step 5-display the output.
Step 6-Stop.

6. Fibonacci Series

Ans:

Step 1: Start
Step 2: Declare variables first term,second term, and temp
Step 3: Initialize variables first term⟵0 and second term⟵1
Step4: Display first term and second term
Step 5: Repeat the steps until second terms< 1000

5.1. temp←second terem
5.2 second term-second_term+first term
5.3 first term=temp
5.4 : Display second_term
Step6: stop.

4. Keywords, Identifier, Constant, Tokens, data types in C.

Ans:

i) Keyword:

Keywords are special words in C programming which have their own predefined meaning. The functions and meanings of these words cannot be altered.
Some keywords in C Programming are if, while, for, do, etc.

ii) identifier:

Identifiers are user-defined names of variables, functions and arrays. It comprises of combination of letters and digits.
Example
int age1;
float height_in_feet;
Here, age1 is an identifier of integer data type. Similarly height_feet is also an identifier but of floating integer data type.

iii) Constant:

Constants refer to fixed values that the program may not change during its execution. These fixed values are also called literals. Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal. There are enumeration constants as well.
Example:
121
234
3.14

iv) Token: tokens in C as the smallest individual elements in a program that is meaningful to the functioning of a compiler.

v) Datatype:

In the C programming language, data types constitute the semantics and characteristics of storage of data elements.

5. Define type casting. Give any one example

Ans: The conversion of one data type to another is known as type casting. The values are changed for the respective calculation only, not for any permanent effect in a program.

For example, x=int (7.5) means 7.5 is converted to integer by truncating it i.e. 7 b=(int) 22.7/(int) 5.3 means 22.7 will be converted to 22 and 5.3 to 5 so answer will be 22/5=4 c=(double) total/num means the answer will be in float value. p=sin((int)x) means x will be converted to integer and then sine angle will be calculated.

6.

UNIT 2: Control Structures

7. if else, nested if else, Operators, while, do while,for, switch case in c.

Ans:

- **Explain nested if-else with example.**

When a series of decision is required, nested if-else is used. Nesting means using one if-else construct within another one. If the condition in the outer if, is true, then only the inner if-else will get executed. Further the statements in the inner if will get execute only if the condition of inner if, evaluates to true. If it is false, the statements in inner else will get executed. If the outer if evaluates to false, then the statements in outer else get executed.

General syntax:

```
if(condition)
{
{ statements }
else
{ statements }
} else
{ statements }
```

Example:

```
#include
#include
void main()
{
int val;
```

```
clrscr();
printf("Enter a number");
scanf("%d",&val);
if(val>=5)
{
if(val>5)
{
printf("Number is greater than 5");
}
else
{
printf("Number is equal to 5");
}
}
else
{
printf("Number is less than 5");
}
getch();
}
```

- **Explain conditional operator with example.**
- Conditional Operator (Ternary Operator):
- It takes the form ? : to construct conditional expressions
- The operator ? : works as follows:
- exp1 ? exp2 : exp 3
- Where exp1, exp2 and exp3 are expressions.exp1 is evaluated first, If it is true, then the expression exp2 is evaluated and becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.
- E.g.
- int a=10,b=5,x;
- x=(a>b) ? a : b;

- **State any two differences between while and do-while statement. (Note: Any 2 points shall be considered).**

while	Do-while
In 'while' loop the controlling condition appears at the start of the loop.	In 'do-while' loop the controlling condition appears at the end of the loop.
The iterations do not occur if, the condition at the first iteration, appears false.	The iteration occurs at least once even if the condition is false at the first iteration.
It is an entry controlled loop	It is an exit controlled loop
while(condition) { body	do { body

- **Give syntax of if-else ladder.**

```
if(condition_expression_One)
{
statement1;
}
else if (condition_expression_Two)
{
statement2;
}
else if (condition_expression_Three)
{
statement3;
}
else
{
statement4;
}
```

- **State any four decision making statement.**

Decision making statement:
1. if statement
2. if-else statement
3. if-else-if ladder
4. Nested if-else statement
5. switch statement
6. conditional operator statement (? : operator)

- **State use of while loop with syntax.**

While loop is used in programming to repeat a specific block of statement until some end condition is met.
The syntax of a while loop is:
while (test Expression)
{
Statements... statements....
}

- **List logical operators in 'C'.**

Logical operators in C include:
1) && (Logical AND): Returns true if both operands are true.
2) || (Logical OR): Returns true if at least one operand is true.
3) ! (Logical NOT): Returns true if the operand is false, and false if the operand is true.

Write the meaning of '&' and * with respect to pointer.

1) & (Address-of Operator): It returns the memory address of a variable.
2) * (Dereference Operator): It is used to access the value stored at the address held by a pointer.

- **Explain any four bit-wise operator used in 'C' with example.**

Bitwise operators:
Bitwise OR-|

It takes 2 bit patterns and performs OR operations on each pair of corresponding bits. The following example will explain it.
1010
1100
OR
1110

Bitwise AND- &
It takes 2 bit patterns and performs AND operations with it.
1010
1100
AND 1000
The Bitwise AND will take pair of bits from each position, and if only both the bit is 1, the result on that position will be 1. Bitwise AND is used to Turn-Off bits.

Bitwise NOT
One's complement operator (Bitwise NOT) is used to convert each "1-bit to 0-bit" and "0-bit tol-bit", in the given binary pattern. It is a unary operator i.e. it takes only one operand.
1001
NOT
0110

Bitwise XOR ^
Bitwise XOR ^, takes 2 bit patterns and perform XOR operation with it.
0101
0110

XOR 0011

Left shift Operator - <<
The left shift operator will shift the bits towards left for the given number of times.
int a=2<<1;

Right shift Operator - >>
The right shift operator will shift the bits towards right for the given number of times.
int a=8>>1;

- **Implement a program to demonstrate logical AND operator. (Note: Any other relevant logic shall be considered)**

```
#include
#include
void main()
{
int i;
int j;
clrscr();
printf("Enter the values of i and j");
scanf("%d%d",&i,&j);
if(i==5 && j==5)
{
printf("Both i and j are equal to 5");
}
else
{
printf("Both the values are different and either or both are not equal to 5");
}
getch();
}
```

- **Explain do-while loop with example.**

Do-While statement:

- In some applications it is necessary to execute the body of the loop before the condition is checked; such situation can be handled by do statement.
- At least once the body of loop will be executed.
- do statement, first executes the body of the loop.
- At the end of the loop, the test condition in the while statement is evaluated. If the condition is true, then it continues to execute bodyof the loop once again.
- This process continues as long as the condition is true.

- **Explain increment and decrement operator.**

Increment operator is used to increment or increase the value of a variable by one. It is equivalent to adding one to the value of the variable. The symbol used is ++.
The decrement operator is used to decrement or decrease the value of variable by 1. It is equivalent to subtracting one from the value of the variable. The symbol used is –.
Syntax:
++var or var++ for increment and --var or var--for decrement.

Example:

```
int m=5;
int n = ++m;
printf(%d%d",m,n);
```

When the increment operator is used prior to the variable name m, the value of the variable m is incremented first and then assigned to the variable n. The values of both the variable m and n here will be 6. But if the increment operator ++ is used after the variable name, then the value of the variable m is assigned to the variable n and then the value of m is increased. Therefore the values of m and n will be 6 and 5 respectively

Example for decrement operator

```
int m=5;
int n=--m;
printf("%d%d",m,n);
or
#include
#include
void main()
{
int m=4,n=6;
clrscr();
printf("values of m and n before changing%d%d",m,n);
m++;
n--;
printf("\nvalues after changing%d%d",m,n);
getch();
}
```

- **Explain conditional operator with example.**

Conditional operators return one value if condition is true and returns another value is condition is false. This operator is also called as ternary operator as it takes three arguments.

Syntax :

(Condition? true_value: false_value);

Example:

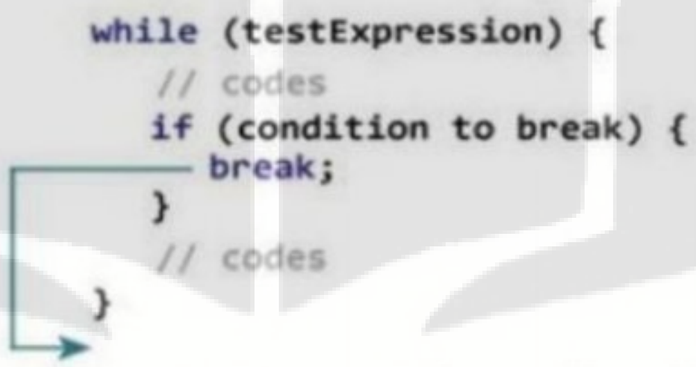
```
#include
#include
void main()
{
int i;
clrscr();
printf("Enter a number:");
scanf("%d",&i);
i%2==0?printf("%d is even",i):printf("%d is odd",i) ;
getch();
}
```

- **Illustrate the use of break and continue statement with example.**
  (Note:- Any other example shall be considered)

Break:

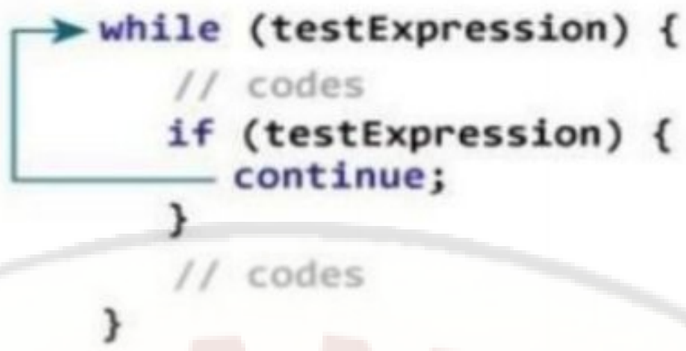
It breaks the execution of the loop which allows exiting from any loop or switch, such that break statement skips the remaining part of current iterations of the loop.

Syntax: break;



Continue: It is used when it is required to skip the remaining portion of the loop without breaking loop it will transfer control directly to next iteration

Syntax: continue;



In given program sequence if "break" executes then execution control will jump out of loop & next statement after loop will be executed. In given program sequence if "continue" executes then execution control will skip remaining statements of loop & will start next iteration of loop

- **Calculate factorial of a number using recursion. (Note: Explanation /algorithm /program shall be considered)**

```
#include
#include
int factorial(int no)
{
if(no==1)
return(1);
else
return(no*factorial(no-1));
}
void main()
{
intfact,no;
clrscr();
printf("\n Enter number");
scanf("%d",&no);
fact=factorial(no);
printf("\n Factorial number=%d",fact);
getch();
}
```

8. Pattern Program Questions thought in lecture.

Ans:

Write a C program to print following pattern using loop

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

```
#include <stdio.h>
void main() {
   int rows = 5;

   for (int i = 1; i <= rows; i++) {
      for (int j = 1; j <= i; j++) {
         printf("%d ", i);
      }
      printf("\n");
   }

   getch();
}
```

Write a C programm to print a pattern.

```
#include <stdio.h>

int main()
{
int r,c;
for(r=1;r<=5;r++)
{
   for(c=1;c<=r;c++)
   {
    printf ("*");
   }
   printf ("\n");
}
```

```
  return 0;
}
```

Output:

```
*
* *
* * *
* *  * *
*  *  * * *
```

Write a C program to print a pattern.

```
#include <stdio.h>
#include <conio.h>
Void main()
int i,j;
for (i=1; i<= 5; i++)
{
  for (j=1;j<= 5; j++)
{
   Prinf("%d", i);
}
  printf("\n");"
  getch();
}
```

Output:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Write a C program to print a pattern.

```
1
2 3
4 5 6
7 8 9 10
```

11 12 13 14 15

```
#include <stdio.h>
#include <conio.h>
int i, j, n=1;
clrscr();
for (i=1, k=5; i++):
for (j=1;j<= 1; j++)E
{
  Printf ("%d", n++);
}
printf ("\n");
}
getch();
}
```

Write a C program to print a pattern.

```
A
B B
C C C
D D D D
E  E  E E  E
F  F   F  F  F  F
```

```
#include <stdio.h>
#include<conio.h>
void main()
{
int i,j;
for(i = 65; i<=70; i++)
{
   for(j=65; j<= i;j++)
{
  Printf ("%c",  i );
}
Printf ("\n");
}
```

```
getch();
}
```

UNIT 3 : Array and Structures

9. Define:

(i) Two dimensional array

Ans:

A two-dimensional array is a collection of elements organized into rows and columns, forming a grid-like structure.

(ii) Multi-dimensional array

Ans:

Multi-dimensional array: An array with more than one dimension is called as multidimensional array. For example, float x[3][4]; Similarly, you can declare a three- dimensional (3d) array. For example, float y [2][4][3];

10. Array Program

1. Write a program to sort elemnts of array in ascending / Descending order.

Ans:

i) Ascending order:-

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

void main()
{
int i,j,n,a,num[30];
printf ("enter the value\n");
scanf("%d",&n);
printf ("enter the numbers\n");
```

```
for(i=0;i<n;i++)
{
scanf("%d",&num[i]);
}
for(i=0;i<n;i++)
{
    for(j=i+1;j<n;j++)
   {
      if(num[i]>num[j])
      {
         a=num[i];
         num[i]=num[j];
         num[j]=a;
      }

}
}
printf("numbers in accendung order");
for (i=0;i<n;i++)
{
printf ("%d\n",num[i]);
}
  getch();
```

ii) Descending order:-

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   int  i ,j ,a ,n ,num[30];
    printf ("enter the value of  number");
     scanf ("%d",&n);
      printf ("enter the number");
      for (i=0;i<5;i++)
       {
for (j=i+1;j<5;j++)
  {
    if (num[i] < num[j])
  }
```

```
    }
      for (i=0;i<5;i++)
       {
           Printf ("the descending order value");
       }
   getch();
}
```

2. Write a program in C to find largest/smallest element from an array.

Ans:

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int arr[100], n, i;
   int largest, smallest;
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);
   printf("Enter %d integers:\n", n);
   for (i = 0; i < n; i++)
{
      scanf("%d", &arr[i]);
   }
   largest = arr[0];
   smallest = arr[0];
   for (i = 1; i < n; i++)
{
      if (arr[i] > largest)
{
         largest = arr[i];
   }
      if (arr[i] < smallest)
{
         smallest = arr[i];
      }
}

printf("Largest element in the array is: %d\n", largest);
```

```
   printf("Smallest element in the array is: %d\n", smallest);
  getch();
}
```

3. Matrix addtion, Multiplication.

Ans:

i) Matrix addition

```
  #include <stdio.h>
  #include<conio.h>
  int main()
  {
    int a[3][3],b[3][3],c[3][3],i,j;
   printf ("enter the values of 1st mat:");
    for(i=0;i<3;i++)
     {
       for(j=0;j<3;j++)
         {
            scanf("%d",&a[i][j]);
         }
     }
printf ("enter the values of 2nd mat:");
    for(i=0;i<3;i++)
     {
       for(j=0;j<3;j++)
         {
              scanf("%d",&b[i][j]);
         }
     }
        for(i=0;i<3;i++)
          {
             for(j=0;j<3;j++)
               {
                 c[i][j]=a[i][j]+b[i][j];
          }
}
   printf ("\n Addition :\n");
```

/

```
for(i=0;i<3;i++)
  {
     for(j=0;j<3;j++)
        {
     printf("%d\t",c[i][j]);
        }
      printf("\n");
  }
getch();
}
```

i) Matrix multiplication

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
 int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
 system("cls");
 printf("enter the number of row=");
 scanf("%d",&r);
 printf("enter the number of column=");
 scanf("%d",&c);
 printf("enter the first matrix element=\n");
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
  scanf("%d",&a[i][j]);
  }
 }
printf("enter the second matrix element=\n");
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
```

```
  scanf("%d",&b[i][j]);
  }
 }

printf("multiply of the matrix=\n");
  for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
  mul[i][j]=0;
  for(k=0;k<c;k++)
  {
  mul[i][j]+=a[i][k]*b[k][j];
  }
  }
 }
 //for printing result
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
  printf("%d\t",mul[i][j]);
  }
  printf("\n");
 }
 getch();
}
```

4. Write a program to declare an array of 5 elements and display sum of all array elements.

Ans:

```
#include <stdio.h>
#include<conio.h>
void main()
{
  int arr[5] = {10, 20, 30, 40, 50};
```

```
    int sum = 0;
    for (int i = 0; i < 5; i++)
{
      sum += arr[i];
   }
   printf("Sum of all array elements: %d\n", sum);
   getch();
}
```

11. Structure Programs:

1. Write a C program to declare structure employee having data member name, age, designation and salary. Accept and display information of 1 employee.

Ans:

```
#include <stdio.h>
#include<conio.h>
struct Employee
 {
   char name[50];
   int age;
   char designation[50];
   float salary;
};

void main()
 {
    struct Employee emp;
   printf("Enter employee information:\n");
   printf("Name: ");
   scanf("%s", emp.name);
   printf("Age: ");
   scanf("%d", &emp.age);
   printf("Designation: ");
   scanf("%s", emp.designation);
   printf("Salary: ");
   scanf("%f", &emp.salary);
```

```
   // Display employee information
   printf("\nEmployee Information:\n");
   printf("Name: %s\n", emp.name);
   printf("Age: %d\n", emp.age);
   printf("Designation: %s\n", emp.designation);
   printf("Salary: %.2f\n", emp.salary);
getch();
}
```

2. Develop a program using structure to print data of three students having data members name, class, percentage. (Note: Any other relevant logic shall be considered)

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
struct student
{
char name[20];
char c[20];
int per;
} s[3];
int i;
clrscr();
for(i=0;i<3;i++)
{
printf("Enter name, class, percentage");
scanf("%s%s%d", &s[i].name,&s[i].c,&s[i].per);
}
for(i=0;i<3;i++)
{
printf("%s %s %d\n",s[i].name,s[i].c,s[i].per);
}
getch();
}
```

3. Write a program to declare structure student having rollno, name & marks. (Note: Any other correct logic shall be considered). Accept and display data for three students.

Unit 4 : Functions

12. String Functions with program example.
(i) strcmp() (ii) strcpy (iii) strlen() (iv) strcat()

(i) strlen()

Ans.

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

int main() {
  char alphabet[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  printf("%d", strlen(alphabet));
  return 0;
}
```

(ii) strcpy ()

Ans.

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

int main() {
  char str1[20] = "Hello World!";
  char str2[20];

// Copy str1 to str2
strcpy(str2, str1);
```

```
  // Print str2
  printf("%s", str2);

  return 0;
}
```

(iii) strcmp()

Ans.

```
#include  <stdio.h>
#include <string.h>
int main() {
  char str1[] = "Hello";
  char str2[] = "Hello";
  char str3[] = "Hi";

  // Compare str1 and str2, and print the result
  printf("%d\n", strcmp(str1, str2));

  // Compare str1 and str3, and print the result
  printf("%d\n", strcmp(str1, str3));

  return 0;
}
```

(iv) strcat()

Ans.

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

int main() {
```

```
char str1[20] = "Hello ";
char str2[] = "World!";

// Concatenate str2 to str1 (the result is stored in str1)
strcat(str1, str2);

// Print str1
printf("%s", str1);

return 0;
}
```

13. Types of User defined Functions.

Ans.

Different Types of User-defined Functions in C

There are four types of user-defined functions divided on the basis of arguments they accept and the value they return:

1. Function with no arguments and no return value
2. Function with no arguments and a return value
3. Function with arguments and no return value

Function with arguments and with return value

14. Recursion

1. Recursion? Program example(factorial)

Ans.

```
#include
#include
int factorial(int num)
{
if(num==1)
{
return 1;
```

```
}
else
{
return(num*factorial(num-1));
}
}
void main()
{
int num;
int result;
clrscr();
printf("Enter a number");
scanf("%d",&num);
result=factorial(num);
printf("Factorial of %d is %d",num,result);
getch();
}
```

2. Recursion ? Advantages and disadvanages?

Ans.

Advantages and Disadvantages of Recursive Functions

ADVANTAGES

- It avoids unnecessary calling of functions.
- It can be used as a substrate for iteration where the iterative solution is very complex.
- It is extremely useful when applying the same solution

DISADVANTAGEs

- It is confusing.
- The exit point of recursive function must be explicitly coded.

It is difficult to trace the logic of function.

15. Call by Value / Call by Reference

1. Difference between Call by Value and Call by Reference.

Ans.

Sr. No.	Call by value	Call by reference
1	When function is called by passing values then it is call by value	When function is called by passing address of variable then it is called as call by reference.
2	Copy of actual variable is created when function is called.	No copy is generated for actual variable rather address of actual variable is passed.
3	In call by value, memory required is more as copy of variable is created.	In call by reference, memory required is less as there is no copy of actual variables.
4	Example:- Function call - Swap (x,y); Calling swap function by passing values.	Example:- Function call – Swap (&x, &y); Calling swap function by passing address.
5	Original (actual) parameters do not change. Changes take place on the copy of variable.	Actual parameters change as function operates on value stored at the address.

2. Call by Value ? Program Example

Ans.

```
#include<stdio.h>
#include<conio.h>
void swap(int a, int b)
{
```

```
int temp;
temp=a;
a=b;
b=temp;
printf("Numbers after swapping no1=%d and no2=%d",a,b);
}
void main()
{
int no1, no2;
clrscr();
printf("Enter the 2 numbers");
scanf("%d%d",&no1,&no2);
printf("Numbers before swapping no1=%d and no2= %d",no1, no2);
swap(no1,no2);
getch();
}
```

3. Call by Reference ? Program Example

Ans.

```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x = 5, y = 10;
    printf("Before swap: x = %d, y = %d\n", x, y);
    swap(&x, &y); // Passing the addresses of x and y
    printf("After swap: x = %d, y = %d\n", x, y);
    return 0;
}
```

16. Storage Classes with Program Example?

UNIT 5 : Pointers

17. Explain '*' and '&' operators in pointer.

Ans.

1) & (Address-of Operator): It returns the memory address of a variable.

2) * (Dereference Operator): It is used to access the value stored at the address held by a pointer.

18. Pointer Programs

1. Write a program find largest number from an array using pointer

Ans.

```
#include<stdio.h>
 // Function to find the largest number in an array using pointer
 int findLargest(int *arr, int size) {
int *ptr = arr;
int largest = *ptr;
for (int i = 1; i < size; i++) {
if (*(ptr + i) > largest) {
largest = *(ptr + i);
 }
 }
return largest;
}
int main() {
int numbers[] = {23, 45, 12, 67, 89, 54, 32};
int size = sizeof(numbers) / sizeof(numbers[0]);
```

```
// Call the function to find the largest number int result = findLargest(numbers, size);
// Display the result printf("The largest number is: %d\n", result);
return 0;
```

2. Pointer Arithmatic

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i = 10;
int *ptr=&i;
clrscr();
printf("%x%d",ptr,i);
ptr++;
printf("\n%x%d",ptr,i);
printf("\n%x",ptr+2);
printf("\n%x",ptr-2);
getch();
}
```

3. Pointer ? Advantages and Disadvantages?

4. Write a program in c to inpute values in array and print in reverse orderr.

Ans.

```
#include<stdio.h>
#include<conio.h>
#define max 50
void main()
{
int a[max],i,n;
clrscr();
printf("\n Enter number of elements:");
scanf("%d",&n);
```

/

```
printf("\n Enter array element:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Array elements in reverse order:");
for(i=n-1;i>=0;i--)
printf("\t%d",a[i]);
getch();
}
```

5. Explain how to pass pointer to function with example.

Ans.

```
#include
int add(int *);
void main()
{
int *ptr,pos=0;
clrscr();
printf("Enter position:");
scanf("%d",&pos);
ptr=&pos;
printf("\nSum=%d",add(ptr));
getch();
}
int add(int *p)
{
int i=0;
int sum=0;
for(i=1;i<=(*p);i++)
{
sum=sum+i;
}
return sum;
}
```

19. Program Practical Questions

1. Write an algorithm and draw a flowchart to find largest number from three numbers.

2. Write a program in C to check whether the entered number is even or odd.

Ans.

```
#include <stdio.h>
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);

   // true if num is perfectly divisible by 2
   if(num % 2 == 0)
      printf("%d is even.", num);
   else
      printf("%d is odd.", num);

   return 0;
}
```

3. Write a program in C to check whether the entered number is positive or Negative.

Ans.

```
#include <stdio.h>

int main() {

   double num;
   printf("Enter a number: ");
   scanf("%lf", &num);
   if (num <= 0.0) {
```

```
    if (num == 0.0)
        printf("You entered 0.");
    else
        printf("You entered a negative number.");
  }
  else
    printf("You entered a positive number.");

  return 0;
}
```

4. Write a function to print Fibonacci series starting from 0, 1. (Note: Any other correct logic shall be considered).

Ans.

```
#include <stdio.h>
int main() {

 int i, n;

 // initialize first and second terms
 int t1 = 0, t2 = 1;

 // initialize the next term (3rd term)
 int nextTerm = t1 + t2;

 // get no. of terms from user
 printf("Enter the number of terms: ");
 scanf("%d", &n);

// print the first two terms t1 and t2
printf("Fibonacci Series: %d, %d, ", t1, t2);
```

```
// print 3rd to nth terms
for (i = 3; i <= n; ++i) {
printf("%d, ", nextTerm);
t1 = t2;
 t2 = nextTerm;
 nextTerm = t1 + t2;
}

return 0;
}
```

5. Write a program for checking weather given number is prime or not.

Ans.

```
#include <stdio.h>

int main() {

 int n, i, flag = 0;
 printf("Enter a positive integer: ");
 scanf("%d", &n);

 // 0 and 1 are not prime numbers
 // change flag to 1 for non-prime number
 if (n == 0 || n == 1)
  flag = 1;

 for (i = 2; i <= n / 2; ++i) {

  // if n is divisible by i, then n is not prime
 // change flag to 1 for non-prime number
 if (n % i == 0) {
  flag = 1;
```

```
    break;
  }
}

// flag is 0 for prime numbers
if (flag == 0)
  printf("%d is a prime number.", n);
else
  printf("%d is not a prime number.", n);

return 0;
}
```

6. Develop a program to accept an integer number and print whether it is palindrome or not. (Note: If string is considered instead of number for palindrome checking, then that logic shall be considered)

Ans.

```
#include <stdio.h>
int main() {
  int n, reversed = 0, remainder, original;
    printf("Enter an integer: ");
    scanf("%d", &n);
    original = n;

    // reversed integer is stored in reversed variable
    while (n != 0) {
      remainder = n % 10;
      reversed = reversed * 10 + remainder;
      n /= 10;
    }

    // palindrome if orignal and reversed are equal
```

```
    if (original == reversed)
        printf("%d is a palindrome.", original);
    else
        printf("%d is not a palindrome.", original);

    return 0;
}
```

7. Write a program to print reverse of a entered string using pointer. (Note: Any other correct logic shall be considered).

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str[10],*ptr;
int 1=0;
clrscr();
printf("Enter string:");
scanf("%s", str);
ptr=str;
while(*ptr!='\0')
{
l=1+1;
ptr ptr+1;
}
while(10)
{
ptr ptr-1;
printf("%c", *ptr);
-1;
}
getch();
}
```

8. Design a program in C to read the n numbers of values in an array and display it in reverse order. (Note: Any other relevant logic shall be considered)

Ans.

```
#include<stdio.h>
#include<conio.h>
#define max 50
void main()
{
int a[max],i,n;
clrscr();
printf("\n Enter number of elements:");
scanf("%d",&n);
printf("\n Enter array element:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Array elements in reverse order:");
for(i=n-1;i>=0;i--)
printf("\t%d",a[i]);
getch();
}
```

9. Area of Circle Program.

Ans.

```
#include <stdio.h>

int main()
{

  float pi = 3.14;
  printf("Enter the radius of the circle: ");
  scanf("%f", &radius);
  // calculating the area
  area = pi * radius * radius;
```

```
    printf("The area of the circle is %f", area);

    return 0;
}
```

10. Switch case program Vowel and Consonant.

```c
#include <stdio.h>

int main()
{
    char ch;

    /* Input an alphabet from user */
    printf("Enter any alphabet: ");
    scanf("%c", &ch);

    /* Switch value of ch */
    switch(ch)
    {
        case 'a':
            printf("Vowel");
            break;
        case 'e':
            printf("Vowel");
            break;
        case 'i':
            printf("Vowel");
            break;
        case 'o':
            printf("Vowel");
            break;
        case 'u':
            printf("Vowel");
            break;
        case 'A':
```

```
            printf("Vowel");
            break;
        case 'E':
            printf("Vowel");
            break;
        case 'I':
            printf("Vowel");
            break;
        case 'O':
            printf("Vowel");
            break;
        case 'U':
            printf("Vowel");
            break;
        default:
            printf("Consonant");
    }

    return 0;
}
```

20. malloc() and calloc()

What is malloc()?

The malloc is also known as the memory allocation function. malloc() dynamically allocates a large block of memory with a specific size. It returns a void type pointer and is cast into any form.

What is calloc()?

The calloc() function allocates a specific amount of memory and initializes it to zero. The function can be cast to the desired type when it returns to a void pointer to the memory location.

S.No.	malloc()	calloc()
1.	malloc() function creates a single block of memory of a specific size.	calloc() function assigns multiple blocks of memory to a single variable.
2.	The number of arguments in malloc() is 1.	The number of arguments in calloc() is 2.
3.	malloc() is faster.	calloc() is slower.
4.	malloc() has high time efficiency.	calloc() has low time efficiency.
5.	The memory block allocated by malloc() has a garbage value.	The memory block allocated by calloc() is initialized by zero.
6.	malloc() indicates memory allocation.	calloc() indicates contiguous allocation.

21. Difference between

1. Array vs Structure

Parameter	Structure in C	Array in C
Definition	A Structure is a data structure that can contain variables of different data types.	An Array is a data structure that can only contain variables of the same data type.
Memory Allocation	Structures do not require the data to be stored in consecutive memory locations.	Arrays store data in contiguous memory locations, meaning that memory blocks are assigned consecutively.
Accessibility	To access elements in a Structure, the name of the specific element is required.	In Arrays, elements can be accessed by their index.
Pointer	Structures do not use internal Pointers.	Arrays use internal Pointers that point to the first element in the array.
Instantiation	An object can be created from a Structure even after its declaration in the program.	Arrays do not allow object creation after their declaration.
Data Type Variables	Structures can include input variables of multiple data types.	Arrays can only include input variables of the same data type.
Performance	Structures can be slower to search and access due to the presence of multiple data types.	Arrays can be faster to search and access due to the absence of multiple data types.
Syntax	struct structure_name{ element type 1; element type 2; . . } variable1, variable2. . .;	data_type array_name[size]

2. while vs do while

Difference Between While and Do-While Loop		
Factor	While Loop	Do While Loop
Syntax	while(condition) {statement(s)}	do {statement(s)} while(condition);
Condition	Condition is checked before statement(s) are executed	Statement(s) are executed at least once before condition is checked
IF/EVEN IF Conditions	If condition is false, statement(s) are not executed at all	Even if condition is false initially, statement(s) are executed at least once
Usage	Used when you want to execute statement(s) only if condition is true	Used when you want to execute statement(s) first and then check condition
Example	while(i<5) {console.log(i); i++}	do {console.log(i); i++} while(i<5);

3. Break vs Continue

Parameters	Break Statement in C	Continue Statement in C
Loop Construct	The break statement allows for an immediate exit from a loop construct.	The continue statement does not provide an exit from a loop construct.
Switch and Loop Statement	The break statement can be used with the switch statement and within for, do-while, and while loops. This means the break statement can occur in both loops and switches.	The continue statement cannot be used with the switch statement, but it can be used within for, do-while, and while loops. This means the continue statement can only occur in loops, not switches.
Control	Upon encountering the break statement, control immediately exits from a loop construct.	Upon encountering the continue statement, control automatically returns to the start of a loop statement.
Function	The break statement causes a loop or switch to terminate a case at the moment of its execution. This means a loop or switch will end abruptly upon encountering a break.	The continue statement does not cause loop termination but leads it into its next iteration. This means a loop will execute all its iterations, even if it encounters a continue statement. The continue statement is used to skip statements that appear after the continue in a loop.
Syntax	It can be denoted as: break;	It can be denoted as: continue;

22. Enumerated Data types

```
1. #include <stdio.h>
2.  enum months{jan=1, feb, march, april, may, june, july, august, september, october, nove
   mber, december};
3. int main()
4. {
5. // printing the values of months
6. for(int i=jan;i<=december;i++)
7. {
```

```
8.    printf("%d, ",i);
9.    }
10.     return 0;
11.}
```