Q. No. 2 Part (i) **Application of OS:-**

a) **Time-Sharing**: used in Airlines, Universities. (as all users want to access server simultaneously).

b) **Real Time**: Real time Traffic Control Systems or in oil refinery (to avert explosion) (produce result within fixed deadline)

c) **Embedded**: Home Appliances such as ovens, washing machines etc (perform specific tasks when turned on)

Q. No. 2 Part (ii) **Responsibilities of System Analyst:**

He is the person who solves a problem, plan solutions, give expert advices. They have extensive knowledge of OS.

① Help architects in design phase and designing system.
② Has technical information and inspects the system
③ Help programmers in making user manuals and assist them.
④ Help developer develop and write code.

Q. No. 2 Part (iii) **Deployment Phase:** Also called as implementation phase. It involves installation and activation of new system.

① **Direct implementation:** Old system is completely discarded and new one is implemented. There's no backup of old system.

② **Parallel implementation:** Both (old and new) systems are used together for some time. This help is analyzing new system and data is not lost.

③ **Phased implementation:** It involves gradual introduction of new system and old one is slowly discarded.

Q. No. 2 Part (iv) **Requirement Gathering:** It is the 1st and most important step of SDLC.

1) **Functional Requirements:** These are the capabilities a ~~user~~ programmer must build into the system and are necessary for operation. They are specified by developer.

Example: A system must shut down in case of cyber attack.

2) **Non-Functional requirements:** These are the capabilities that specify the criteria for judgement of the new system. They improve the System, but are not must. Specified by users.

Example: A program must take 3 seconds to open.

Q. No. 2 Part (v)

Output :

5 3 5

Explanation :

initially i=3 . when j=i++ executes . first j is assigned value of 3 Then i is incremented To 4 So, i = 4 and j = 3 .

Then k=++i is executed . No first i is incremented to 5 and then assign to k .

So i = 5 and k = 5

Q. No. 2 Part (vi)

It will print numbers from 6 to 105 ie value of m

OP:

~~6 7 8 9 10 11 - 12 105~~

$\boxed{6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ ----\ 105}$

Running :

When c=0 m is initially 5 then incremented to 6 and output. this continues till c=99 when m becomes m = 105

```
m=5 , c=0
while ( C< 100)
{ m=m+1;
  c=c+1;
}
cout <<m;
```

c=0 m=6
c=1 m=7
c=2 —— 8
c=3 —— 9
c=4 —— 10
c=5 —— 11
6 —— 12
7 —— 13
8 —— 14
9 → 15

Q. No. 2 Part (vii)

```
#include <iostream.h>
#include <string.h>
using namespace std;
int main()
{ char str[20];
  cout<<"Enter a String :";
  cin.get ( str, 20); ⟶ input
  cout<<"Number of characters in string:"<<strlen(str);
                                              ↓
  return 0;                            no. of characters.
}
```

Q. No. 2 Part (viii)

int arr [8]

a) No.of elements : 8 elements

b) Highest index : 7 as arr [7]

c) lowest index : 0 as arr [0]

arr[0] , arr [1] , arr [2] , arr [3] , arr[4] , arr[5] , arr[6] , arr[7]

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

1st element 2nd 3rd 4th 5th 6th 7th 8th

* index starts from 0 so that of last element is $n-1$

Q. No. 2 Part (ix) **Strings:** Strings are a sequence of characters used to store data such as names, addresses etc. They are stored like one dimensional array. It has data type **char** and a **Null Character** is automatically appended.

Decleration: To declare a string; its datatype, name of string and size needs to be defined.

Example: char str[20];

Syntax : data-type string-name [string-size];

Q. No. 2 Part (x) **Modes of file opening:** These are The optional parameters given to open a file in specific mode.

	modes	Purpose
1)	ios : : in	Open file for input (Reading purpose)
2)	ios : : out	Open file for output (Writing purpose)
3)	ios : : bin	Open file ~~for~~ in binary mode)

Example :

```
myfile.open ("test.txt", ios::bin)
```

- Multiple modes can be used using bitwise (|) operator.

Q. No. 2 Part (xi)

Array	Simple variable
1) Used to store data of same type in contigious memory locations	Used to store a single value / character in single portion of memory.
2) They can be of type : int and float	They can be int, float, char.
3) Can store multiple values so saves program space.	Takes more space due to seperate initializations
4) They make programming efficient and help solve complex problems	They are helpful, but less effective than arrays.

5) **Example :** (initialize / declare)
int arr[5] = {1, 2, 6, 4, 7};
float a[20];
	Example:
	int a = 5, b = 6;
	char ch, float p, a, m;

Q. No. 2 Part (xii) **Data Hiding:** Data Hiding in classes or also called Encapsulation is to protect data (data members and member functions) from unauthorized access or give it right to be used in certain places of program or classes. For this purpose we use **Access Specifiers** (they specify access of members). They have 3 types: **Public** (used anywhere), **Private** (only used in class or friend function), **Private** (used in the class or derived class only)

Example :

```
class  Rectangle()
{ Private :
  int a,b,c; → Access within class
  Public:
  int Area() → can be accessed in main().
}{}
```

```
int main()
{ Rectangle R;
  R.Area(); (Valid)
  R.a; (invalid)
```

Q. No. 2 Part (xiii)

a) **Size of Array :** It tells the number of spaces / location reserved for array in the memory. It is enclosed in [ ]

b) **Name of Array :** Name of array is a valid variable name just like name of a simple variable. with which it can be called or used.

c) **Index :** It is also called subscripte or power. It's tells the memory location of each element. It always starts with zero. So last elements index is n-1.

name size

Example : int arr [5] = {1, 6, 2, 8, 9};

index → arr[0] arr[1] a[4] —has index 4

Q. No. 2 Part (xiv) **Roject Monager:** He is an expert who conducts meetings to analyze requirements, creates reports and works to implement them. He has knowledge of programming too.

Role of Project Manager:

1) Manages the Stakeholders of project and identifies them.
2) Manages risks related to system.
3) manages conflicts related to project
4) manages schedule of project
5) manages efficiency of project
6) Conducts meeting and analyzes user requirements

Q. No. 3 (Page 1)

Program to find Area and Perimeter of Square

```
#include <iostream.h>
#include < conio.h >
using namespace std;
int main( )
{ int l;
   cout<<"Enter length of side of square:"<<endl;
   cin>>l;
   cout <<"\nArea of Square = "<<(l*l);
   cout<<"\n Perimeter = "<<(4*l);
   getch();
    return0;
}
```

(**Formulae:**
Area = l*l , Perimeter = 4l)

Q. No. 3 (Page 2)

Q. No. 3 (Page 3)

Q. No. 4 (Page 1)

Variables	Constants
1) Their values change during execution	They have fixed values that do not change.
2) They have special rules for naming them. For instance; special symbol except '_' can't be used or they can't start with numbers.	They don't have special naming rules and can have any fixed value.
3) They have variable size depending on type: int, char, float	Constants can be of any type and size too: int, char, float.
4) They can be initialized or declared	String constants are enclosed in double quotes and characters in single quotes.

5) **Example:** int a, b;
float H;
	Example: int a=2;
	float pi=3.142;

const Qualifier:

With their use, the variable does not remain a variable. It becomes a character constant. It has following syntax.

const datatype variable initialization;

Example:

```
const int AGE = 16;
const float HEIGHT= 8.0;
```

They are usually written in capital letters.

Q. No. 4 (Page 2)

Program:

```
#include<iostream.h>
#include<conio.h>
using namespace std;
int main()
{ int base; → Variable
  const int HEIGHT=6; → constant:
  cout<<"Enter base of tringle:<<endl;
  cin>>base;
  cout<<"\nArea ="<<(base*HEIGHT)/2
  getch()
}
```

Explanation:

When we run this program, we can always change the base, but throught the program value of height remains same and constant.
Hence base is a variable and height is constant.

Q. No. 4 (Page 3)

Q. No. 5 (Page 1)

Continue Statement:

It is used to take the control of to The beginning of the loop if a particular condition is met.

Example:

```
#include <iostream.h>
#include< conio.h >
using namespace std;
void main()
{ int i;
    for(i=0; i<10; i++)
   {if (n>3 && n<7)
     continue;
     cout << i <" " ;  }
     getch();
     return 0;
}
```

Output :-

```
0 1 2 3 7 8 9
```

Explanation:

Number from 3 to 7 ie (4,5,6) are skipped as the condition n>3 and n<7 remains True control continually goes to beginning of loop.

Q. No. 5 (Page 2)

Exit () function:

It is used to terminate the program before its normal termination and return the control to the operating system.

library / header file:

It require stdlib.h header file to be included in the program.

Codes:

It has codes 0 and 1

- ★ Exit (0) return without an error
- ★ Exit (1) indicates that an error might have occured.

Example (code segment):

```
if (n % 2 == 0)
 cout << " It is Even number.";
else
  exit (0);
```

Explanation :-

It would execute the condition. It its true, it will print the message otherwise the program would terminate without giving error.

Q. No. 5 (Page 3)

Q. No. 6 (Page 1)

~~To check if a Number is Prime or Composite~~ ~~(without flags)~~

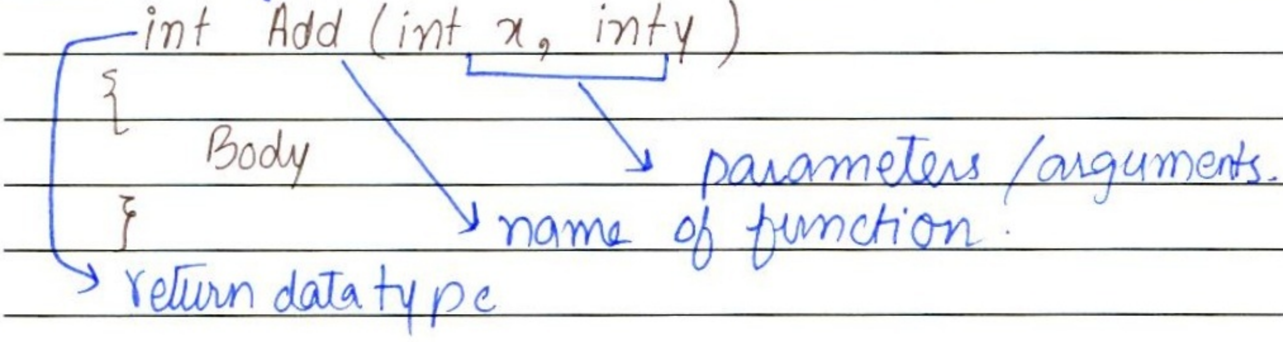
```
#include < iostream.h>
#include < conio.h >
using namespace std;
void main ( )
{ int num;
  cout << "\n Enter a positive Number: " ;
  cin >> num;
  for (int i=2 ; i<=num/2 ; i++)
  { if ( num % i = = 0)
      cout << "\nIt is a Composite number." ;
    else
      cout << "\n It is a Prime number." ;
  }
    getch();
    return 0;
}
```

FUNCTION SIGNATURE

It is the part where we define function and its parts. It has 3 parts

1) return type
2) function name
3) function parameters and their types

Example:



```
int Add (int x, inty )
{
    Body
}
```

* They exist within User defined function.

Advantages of Function:

They help to use and access program in more modularized way. They have many advantages

① Modifying a program becomes easier. We can change function without affecting body of main()

② Any change or update can be done easily without effecting entire program.

③ As program is broken into small chunks so testing becomes easier.

④ We don't need to write entire code again and again. We just define a function and it can be used with a single call. This helps to save space and time.

Q. No. 6 (Page 3)