Q. No. 2 Part (i)

PROCESS	THREAD
1) A process is a program in execution.	1) Thread is a subset of process.
2) Process run in separate memory spaces.	2) Threads run is shared memory spaces.
3) Process is independent.	3) Thread is dependent
4) Any change in process doesn't affect other process.	4) A change in thread may affect other threads of the process.

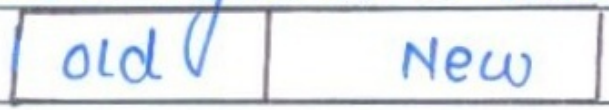
Q. No. 2 Part (ii)

Protection System:

Protection system is important in operating system as it protects the computer from unauthorized access. It creates accounts for users and assigns priveleges to them. This prevents misuse of computer resources. Each user has a login ID which consists of username and password to access the system.

Q. No. 2 Part (iii)

- DIRECT IMPLEMENTION: The old system is completely taken down and new system is used directly.



- PHASED IMPLEMENTATION:- The new system is introduced gradually to learn about any errors. The old system is taken down gradually.

- PILOT IMPLEMENTATION:- The new system is deployed for a few employees/users to learn & use. Then if there are no problems, the new system is completely implemented.

Q. No. 2 Part (iv)

if - else - if	switch
• It tests for equality as well as logic.	• It only tests for equality.
• It can process character, integer and logical expression.	• It can only process character or integer values.
• It allows between multiple statements for multiple choices.	• It uses single statement for multiple choices.
• If no condition is true, else statement is executed.	• If no case matches, default statement is executed.

Q. No. 2 Part (v)

```
int i = 3;
int j = i++;
int k = ++i;
cout << i << " " << j << " " << k;
```

Output :-

5 3 5

Q. No. 2 Part (vi)

```
int  m=5 , c=0;
while (c<900)
{   m = m+1;
    c = c+1;
}
cout << m.
```

Output: **104** (is the output of above program).

6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
29 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 43 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103
104

Q. No. 2 Part (vii)

```
int arr[3][4] = { { 12, 0, 5, 10 }, { 7, 8, 19, 30 },
                  { 33, 1, 2, 22 } };
arr[1][2] = 50;
```

The statement arr[1][2] = 50; replaces 19 with 50.

Q. No. 2 Part (viii)

Access	Public	Private
• Access of members in same class.	Yes	Yes
• Access of members in derived class.	Yes	No
• Access of members out-side the class.	Yes	No.

Q. No. 2 Part (ix)

String :. A string is a sequence of characters. It's data type is 'char'. A string can be a name, title, place, sentence etc. A string is written in double quotes.

To declare a string, we use array.

The general syntax is;

```
char string_name[string_size];
```

where char is the character data type, string_name is any valid variable name, string_size is the number of characters in a string.

All strings end with a null character '\0', therefore string size should be one more than the no. of characters.

e.g `char weekday[20] = "Tuesday";`

Q. No. 2 Part (x)

Modes of file opening:

- ios::in => open file for input operations / read a file. e.g myfile.open("cf.txt","ios::in");
- ios::out => open file for output operations / write a file. e.g myfile.open("cs.txt","ios::out");
- ios::binary => open file in binary mode. e.g myfile.open("cf.txt","ios::binary");

To open a file with a mode, the mode is written as argument of the open() function.

i.e;

myfile.open("file_name","mode");

Q. No. 2 Part (xi)

Reference Operator (&):

The reference or address operator is an ampersand sign (&). It is used in pointers to point to the address of variable. As we know that a pointer variable is used to point to the memory address of another variable. To do this, reference operator is used.

for example;

```
float temperature;
float *Ptemperature; // pointer variable
Ptemperature = &temperature;
cout << Ptemperature;
```

output;

2x5362ccx (address of temperature variable)

Q. No. 2 Part (xii)

Data Hiding :- Data hiding is used in class to prevent illegal access to the class. C++ provides data levels to protect data. Protected class members can be accessed by the members of the same class or friend function. Private members can only be accessed within the class, public members can be accessed from class, derived class or main() function:

Example:

```
class A
{ private:
    int x, int y:
  public:
    int area()
    { return (x*y);
    }
};
void main()
{ A a1;
  a1.area(5,10);
}
```

private ← int x, int y:

public function ← int area()

Q. No. 2 Part (xiii)

a) Size of array:. Size of array is the number of elements in an array. It is written in square brackets while declaring array.

b) Name of Array:. Name of array is a valid variable name that identifies an array.

c) Index :. Index or subscript of array identifies the element of array. Index of array always starts at 0. An array a[5] will have index a[0], a[1], a[2], a[3], a[4]

```
int arr[6];
    |   |  |
    |   |  -> size
    |   -> name
```

Q. No. 2 Part (xiv)

PROJECT MANAGER :.

A project manager is responsible for the planning, creation and development & closing of a project. He/she has extensive knowledge in programming & management skills. A project manager is responsible for

- Planning the project budget.
- managing the project risks.
- managing the project conflicts.
- managing the project tasks.
- managing the project members
- managing the project stakeholders.

Q. No. 3 (Page 1)

PROGRAM

```
#include <iostream.h>
#include <conio.h>
using namespace std;
void main()
{   int l, w, p, area;
    float area;
    cout << "Enter length of square" << endl;
    cin >> l;
    cout << "Enter width of square" << endl;
    cin >> w;   // l = w for square.
    area = l * w;
    cout << "The Area of square is = " << area;
    p = 4 * l; // as for square l = w;
    cout << "Perimeter of square is = " << p;
    getch();
}
```

On next page.

Q. No. 3 (Page 2)

PROGRAM

```
#include <iostream.h>
#include <conio.h>
using namespace std;
void main()
  {   int  x, area, p;
      cout << " Enter length of square: " << endl;
       cin >> x;
       area = x * x;
     cout << " Area of square = " << area << endl;
     p = 4 * x;
     cout << " Perimeter of square = " << p << endl;
  getch();
  }
```

Output:.

Enter length of square : 2
Area of square = 4
Perimeter of square = 8

Q. No. 3 (Page 3)

Q. No. 4 (Page 1)

PROGRAM

```
class Test
    {
      private:
      int x, y;
      public: int a, b;
     Test ( )
     {
         a = 100;
         b = 100;
     }
     int avg()
     {
         return (a+b)/2;
     }
};
void main ()
   {  Test T1;
    cout << T1.avg ();
   getch();
   }
```

Output:

100

Q. No. 4 (Page 2)

Q. No. 4 (Page 3)

Example of function:

```
#include <iostream.h>
int f1() -> function definition
{
    cout<<"I am a function";
}
void main()
{
    f1(); -> function call.
    getch();
}
```

Output:

I am a function.

FUNCTION

Functions:

Functions are an important feature of C++. A function is a group of statements that together perform a task. There are many advantages of using functions in programs. These are;

- Functions make it divide the program into logical blocks. It becomes clear and easy to understand
- Functions can be individually tested.
- Functions prevent re-writing the same code. We can write the code for function once in a program and call it whenever needed.
- It is easy to change the code in function, instead of changing the whole program.
- In case of modification, functions can be easily modified.

Functions provide modularity in programs. They can be uses defined or built-in.

=> Function Definition:

Function definition defines what the function is supposed to do. It contains the actual code of the function. Function Definition consists of function name, perimeters and block of statements.

=> Function: Call:

To use a function, it is called in the main() function. Then control is transferred from the main() function to the called function and it is executed, then control is again transferred to main() function.

Q. No. 5 (Page 3)

Q. No. 6 (Page 1)

PROGRAM

```
#include <iostream.h>
#include <conio.h>
using namespace std;
void main()
  {
     int  n, flag=0, c;
  cout<<" Enter a positive integer:"<<endl;
   cin >>n;
 if (n>0)
  {  for (c=1; c<=n; c++)
       {   if (n%c==0)
             flag++;
       }
    if (flag==2)
     cout<<" It is a prime number."<<endl;
    else
    cout<<" It is a composite number."<<endl;
  }
else
  cout<<" you entered a negative integer or 0";
  getch();
}
```

Output:

Enter a positive integer : 2

2 It is a prime number.

Q. No. 6 (Page 2)

Q. No. 6 (Page 3)

6 7 8 4 1 2 3
9 10 11 12 13 14 15 16 17 18 19
↳ 10

20 30 4 40 99
25 35 95 104.

2+2+2+2/8

m = 5 , c = 0

while (c < 100) c=99 0 1 2 3 99

{ m = m+1 6 7 8 9 10 11
c = c+1 ; m = 104

}

cout << m ; 6+99 = 105.

6
7
8
9
10
11
12
13
14
15
.
.
.