Q. No. 2 Part (i) DIFFERENCE

`void *ptr1;`	`int *ptr2;`
① It is a pointer variable declared with return type void and can point to an object of any datatype	① It is a pointer variable declared with integer return type and can point to an object of only int data type.
② It can store the address of integer, float or any other type of variable	② It can only store the address of integer type variable
③ Eg: { int n; float a; void *ptr1 = &n; void *ptr1 = &a; } The compiler decides at run time that address of which variable is to be assigned	③ Eg: { int b; float a; ~~void~~ int *ptr2 = &b; // int *ptr2 = &a (invalid) }

Q. No. 2 Part (ii) PROGRAM To Determine whether a given number is divisible by 7 or not:

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main()
{ int a;
  cout << "Enter a number:" << endl;
  cin >> a;
  if (a%7==0)
   { cout << "The entered number is divisible by 7";
else
   cout << "The entered number is not divisible by 7";
return 0;}
```

output

```
Enter a number = 7
The entered number is divisible by 7
```

Q. No. 2 Part (iii) **FUNCTION OVERLOADING:**

Function overloading is a feature of C++ that allows multiple functions that have same name to be used in a program but they must have either different number of parameters or different datatypes of parameters.

ADVANTAGES:

1. Using function overloading, multiple functions with the same names can be declared that have slightly different purposes
2. Function overloading can significantly lower the complexity of programs
3. It increases the readability of programs.
4. It exhibits the behaviour of polymorphism.
5. As multiple functions have same names, therefore, remembering them is easier as compared to remembering more names.

Q. No. 2 Part (iv) **PROGRAM TO PRINT SEQUENCE : 2 5 8 11**

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main ()
{   int a;
    for (a=2; a<=11; a=a+3)
    { cout << a << "   ";
    }
  return 0;
}
```

Output

2 5 8 11

Q. No. 2 Part (ix) **FILE OPENING MODES:**

ios :: in = Open for input operations (Reading a file)
ios :: out = open file for output operations (Writing a file)
ios :: binary = Open in binary mode
ios :: ate = Open file for output operations and moves the read/write control to the end of the file.
ios :: app = Append mode. All the output to the file be appended at the end of the file.
ios :: trunc = If the file is opened for output operations, its existing contents are deleted and replaced with the new one.

Q. No. 2 Part (v) **CORRECTION OF ERRORS:**

```
int a[10]; (a statement must end with a semicolon not colon)
a[10] =30; (In order to assign 30 to the index 10 we should
a[1] =20;   write 10, there is no index with floating point or
            decimal numbers.
            (we cannot use negative numbers as index.
             index should be positive integer value).
```

Q. No. 2 Part (vi) CONVERSION FROM FOR TO do-while loop:

```
{
    int i=1;
    do {
        cout<<i;
        i++; } while (i<=10);
}
```

Q. No. 2 Part (vii) INVALID VARIABLE NAMES:

3xy = A variable must start with an alphabet or underscore. It cannot start with a digit.

void = Reserved words like int, void, float etc cannot be used as variable names.

your(age) = Special characters like $, brackets () cannot be used in variable names.

Q. No. 2 Part (viii) DIFFERENCE

CONSTRUCTORS	DESTRUCTORS
① Constructors are member functions that automatically initializes an object when it is created.	① Destructors are created when an object is destroyed. It fulfills opposite functionality of constructor and thus de allocates the memory already allocated to an object when it is created.
② Constructors can take arguments	② Destructors cannot take arguments
③ Constructors can be overloaded	③ Destructors cannot be overloaded.
④ Constructor donot begins with any special symbol	④ Destructors begins with a tilde(~) symbol
⑤ Eg: class A { } { public: A() {}; (A is a constructor)	⑤ Eg: class B { public: ~B() {}; (B is a destructor).

Q. No. 2 Part (x) DIFFERENCE

Pre-test loop	Post-test loop
① It is while loop and condition is checked at the beginning of the loop.	① It is do-while loop and condition is checked at the end of the loop
② It is entry controlled loop	② It is exit controlled loop
③ It never executed if the condition is false at the beginning of the loop	③ It executes at least once even if the condition is false
④ It does not ends with semicolon	④ It end with a semicolon
⑤ Eg: { int i=1; while (i<=2) { cout<<i<<" "; i++} Output: 1 2	⑤ Eg: { int i=1; do { cout<<"I am in dowhile loop"; cout<<endl; i++}while(i!=2) Output: I am in dowhile loop.

Q. No. 2 Part (xi) Cin.get() And Cin.

Cin.get( ) function is used to read a string that may contain blank spaces. Eg:

```
{ char str[50];
  cout<<"Enter a string:";
  cin.get(str,50);
}
```

Output = Enter a string: Information technology

The same task can be achieved by cin but it have a limitation that it ~~Contains~~ assumes space as a terminating character. That is why cin.get() is preferred.

```
Eg: { char str[50];
      cout<<"Enter a string:";
      cin>>str;}
```

Output: Enter a string: Information

Q. No. 2 Part (xii) PURPOSE OF ASTERISK (*)

`c=a*b;` = In this statement asterisk is used for multiplication. It calculates product of a and b and stores it in c.

`float *p=&a;` In this statement asterisk is used to declare a pointer variable p. It assigns address of a to p.

`*p=90;` = In this statement asterisk is used as dereference operator. It assigns 90 value to the variable where address is stored in p.

Q. No. 2 Part (xiii) RESPONSIBILITIES OF SYSTEM ANALYST:

A system analyst is a professional of software development that studies the problems, plans solutions, recommends software system and coordinates software development to meet business and other requires. He has expertise in a variety of programming languages, operating systems and computer hardware platforms.
His responsibilities includes the following:

1. Plan a system flow
2. Manage system testing
3. Define technical requirements
4. Interact with customers to learn and understand requirements that are then used to produce business requirement documents.
5. Help programmers during system development process
6. Interact with designers to understand software limitations.
7. Document requirement and contribute to user manuals.

Q. No. 2 Part (xiv) CONVERSION FROM CONDITIONAL TO IF-ELSE STATEMENT:

```
{if (a>0)
    s=a*a;
  else
    s=a*a*a;}
```

Q. No. 3 (Page 1) **STATES OF PROCESS:**

A process is a program in execution. It has following five states:

NEW STATE:

This is the first state of a process. Any new service or operation requested by the program to be executed or performed by the process is called new state of process.

Ready State:

A process is said to be in ready state when it is ready for execution but it is waiting to be assigned to processor by operating system.

RUNNING STATE: A process is said to be in running state when it is being executed or it is under execution. A process is assigned to the processor for execution by operating system.

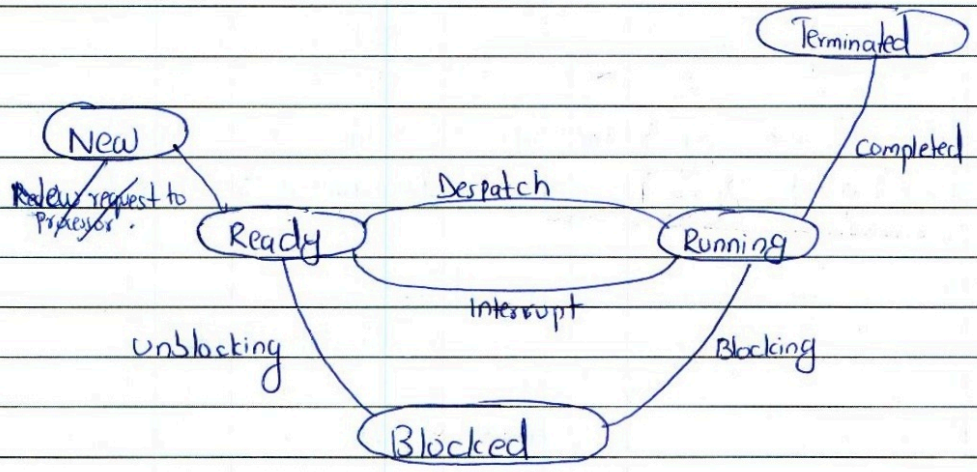
BLOCKED / WAITING STATE:

A process is said to be in blocked or waiting state when it is not under execution. It is waiting for a resource to become available.

TERMINATED STATE:

A process is in terminated state when it compeletes its execution.

Q. No. 3 (Page 2) **DIAGRAM OF VARIOUS STATES OF PROCESS:**



PROCESS AND THREAD:

- A process is a program in execution. It is a dynamic instance of a program
- A thread is a basic ordered sequence of instructions within a process that can be executed independently. A thread is made up of one or more process. Every process has atleast one thread. Multiple threads can also exist.

(on next page)

Q. No. 3 (Page 3)

PROCESS	THREAD
(1) A process is an executing instance of a program	(1) A thread is a subset of a process.
(2) It has its own copy of the data segment of its parent processes	(2) It has direct access to the data segment of its process.
(3) Process runs in separate memory spaces.	(3) Threads run in shared memory space.
(4) Any change in a process doesnot affect other processes	(4) Any change in a thread affects the behaviour of other threads
(5) Processes are controlled by operating system	(5) Threads are controlled by the programmer.
(6) Process is independent.	(6) Threads are dependent.

Q. No. 4 (Page 1)

FEATURES OF OBJECT ORIENTED PROGRAMMING

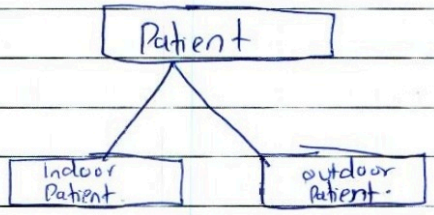
Object oriented Programming enables code reusability in different classes with the help of inheritance and it provides multiple ways of function usage with the help of polymorphism. A detailed explanation of both the features is given below:

INHERITENCE:

A key feature of C++ that leads to object oriented programming is inheritence through which new classes can be created from the already existing classes. Inheritence uses the concept of parent and child class. A parent class is a class from which new classes are created. It is also called base class. A child class is a class which inherits the features of base class. It is also called sub class or derived class.

EXAMPLE:

Consider a patient class which is the base class and it may consists of outdoor patient and indoor patient as derived classes. Both the classes will have certain feature in common such as name, age e.t.c. Outdoor patient will have next day of visit, visit charges et.c as unique features and indoor patient will have Bedno, ward no. as unique features. It can be explained by following diagram:



Q. No. 4 (Page 2) CODE EXAMPLE

```
#include <iostream.h>
#include <conio.h>
using namespace std;
class A
{ public:
   void display()
      { cout<<" I am in base class"<<endl;}
};
class B: public A
 { public:
    void show()
      { cout<<" I am in derived class "<<endl;}
};
int main()
 { B obj;
    obj.display();
    obj.show();
    return 0;}
```

OUTPUT:

I am in base class
I am in derived class

(function of both classes can be called from derived class using inheritence).

POLYMORPHISM:

Polymorphism is the ability to use an operator or functions in multiple ways. It provides different meanings and functionalities to an operator or function. Poly means 'many' and signifies that an operator or function can be used in many ways. It is the use of single operator or functions in various ways. It can be achieved through following methods:

Q. No. 4 (Page 3) **FUNCTION OVERLOADING:** Using multiple functions with same name that have different number or data types of parameters. It exhibits the behaviour of polymorphism.

OPERATOR OVERLOADING: It is the use of single operators for mutiple purposes. For example an operator '+' can be used for addition of two integer number and it can also be used for concatenation of two strings as described below:

25 + 30

"Computer" + "Science"

VIRTUAL FUNCTION OR METHOD: It is a function whose behaviour can be overridden within an inheriting class by a function with same signature. It is of great importance for performing polymorphism in object oriented programming.

DAILY LIFE EXAMPLES:

INHERITENCE:

① A child inherits some characteristics of parents such as eyes like mother and hair like father.

② A new model of car consists of some features of old model such as break, navigation system, tyres e.t.c.

POLYMORPHISM:

A person can behave in many patterns. For eg:

① He can be a student and at the same time friend to another person.

② A person can be an engineer and at the same time He can also be a teacher.

Q. No. 5 (Page 1)

Program To Represent An Alphabet As Vowel Or Consonant And As Lowercase Or Uppercase:

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main()
{ char ch;
  cout << "Enter a character: ";
  cin >> ch;
 if((ch=='a') || (ch=='e') || (ch=='i') || (ch=='o')
    || (ch=='u') || (ch=='A') || (ch=='E') || (
  { cout << "\n" << " The entered character is a vowel" << endl;
  }
  else
     cout << "\n" << " The entered character is a consonant" << endl;
 if((ch>='A') && (ch<='Z'))
    cout << " The entered character is uppercase" << endl;
 else if ((ch>='a') && (ch<='z'))
    cout << " The entered character is a lowercase letter" << endl;
  return 0;
}
```

Q. No. 5 (Page 2)

Output:

Enter a character : A
The entered character is a consonant.
The entered character is upper case.

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main()
{ char ch;
  cout << " Enter a character:";
  cin >> ch;
if ((ch >= 'A') && (ch <= 'Z'))
  { cout << " The entered character is uppercase." << endl;
    if ((ch == 'A') || (ch == 'E') || (ch == 'I') || (ch == 'O')
        || (ch == 'U'))
      cout << " The entered character is a vowel";
      else
       cout << " The entered character is a consonant";
  }
  else if((ch >= 'a') && (ch <= 'z'))
  { cout << " The entered character is lower case." << endl;
    if((ch == 'a') || (ch == 'e') || (ch == 'i') || (ch == 'o')
        || (ch == 'u'))
      cout << " The entered character is a vowel.";
    else
      cout << "The entered character is a consonant.";
  }
return 0; }
```

Q. No. 5 (Page 3)

Output:

Enter a character: A
The entered character is upper case
The entered character is a vowel.

Q. No. 6 (Page 1)

Program To Display Numbers In Reverse Order And Display Their Average:

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main()
{  int a[5], sum=0, avg;
   cout<<" Enter five numbers with space: "<<endl;
   int i;
   for (i=0; i<5 ; i++)
    {  cin>>a[i]; }
   for  sum = sum+a[i]; }
   for (i=4; i>=0; i--)
      {  cout<<i<<" ";
         cout<<endl;}
   avg= sum/5;
   cout<<" Average of 5 numbers= "<<endl;
   return 0;}
```

Output:

Enter five numbers with space :
1 2 3 4 5

(on next page).

Q. No. 6 (Page 2)

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main()
{
   int a[5], sum=0, avg, i;
   cout<<" Enter five integers with space "<<endl;
   for(i=0; i<5; i++)
    { cin>>a[i];
        sum = sum + a[i];
    } cout<<endl; cout<<" The numbers in reverse order "<<endl;
  for(i=4; i>=0; i--)
    {
      cout<<a[i]<<" ";
    } cout<<endl;
   avg= sum/5;
    cout<<" The Average of entered numbers is= "<<avg;
    return 0;
}
```

OUTPUT:

```
Enter five integers with space:
1 2 3 4 5
The numbers in reverse order:
5 4 3 2 1
The Average of entered numbers is= 3
```

Q. No. 6 (Page 3)

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main()
{
    int a[5], sum=0, avg, i;
    cout << "Enter five integers with space: " << endl;
    for (i=0; i<5; i++)
    {
        cin >> a[i];
        sum= sum+ a[i];
    }
    cout << endl;
    cout << "The numbers in reverse order is: " << endl;
    for (i=4; i>=0; i--)
    { cout << a[i] << " "; }
    cout << endl;
    avg= sum/5;
    cout << " The Average of entered numbers is = " << avg;
    return 0;
}
```

OUTPUT:

Enter five integers with space:
1 2 3 4 5
The numbers in reverse order is:
5 4 3 2 1
The Average of entered numbers is = 3