S1612037.

Experiment No 2.

e: Complex numbers operator overloading.

m: Implement a class complex which represents the complex No. data type. Implement the following operations.

1) Constructor (including a default constructor which ~~ext~~ creates the complex number 0+0i).
2) Overloaded operator + to add 2 complex nos.
3) Overloaded operator * to multiply 2 complex nos.
4) Overloaded << and >> to print and read complex numbers.

quisites:- To Fundamentals of Programming Languages-I and II

ective:- To learn the concept of constructor, default constructor, copy constructor, parameterised constructor's operator overloading using friend function.

eory:- Operator overloading:

It is a specific case of polymorphism where different operators have different implementations depending on their arguments.

In C++ the overloading principle applies not only to functions, but to operators too. That is, of operators can be extended to work not just with built-in types but also classes. A programmer can provide his or her own operator to a class

by overloading the built-in operators to some specific computation when the operator used on objects of the class.

You can redefine or overload most of the built operators available in C++. Thus a program can use operators with user-defined types as

Overloaded operators are functions with special names the keyword operator followed by the symbol for the operator being defined. Like other function, an overloaded operator has return type and a parameter list.

```
Box operator + (const Box &);
```

declares the addition operator that can be used add 2 Box objects and returns final Box ob

1. Unary Operator.
The unary operators operate on a single ope and following are the examples of Unary operators. The increment (++) and decrement operators. The unary minus (-) operator. The logical not (!) operator.

② Binary Operator:-

Overloading with a single parameter is called binary operator overloading. Similar to unary operators, binary operators can also be overloaded. Binary operators require 2 operands, and they are overloaded by using member functions & friend functions.

Overloadable / Non-overloadable Operators:-

Following is the list of operators which can be overloaded.

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new[]	delete	delete[]

Following is the list of operators which cannot be overloaded.

:: , .* , . , ?:

Utilities:- Linux Operating Systems, GCC, Eclipse framework, VC++.

Algorithm :-

1. Start.
2. Read 2 complex numbers.
3. Perform arithmatic operations on 2 complex nu
4. Print the output.
5. Stop.

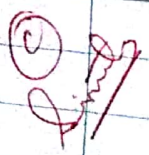
Input :- Complex numbers with real and imaginary no for 2 complex numbers.

Example: Complex No 1 : Real part : 5
Imaginary part : 4
Complex No 2 : Real part : 5
Imaginary part : 4

Output :- Numbers after performing arithmatic operation following format :-

Operation Performed : Complex No. 1
Operation Symbol: Complex NO : 2
RESULT .

Conclusion :- Hence, we have studied concept of operat overloading.



```
/*NAME:- Vrushil Soni
CLASS:- S.E(C)
ROLL No:- S1612037
SUB:- OOP
                              Assignment No:- A_1 (Complex No)

Implement a class Complex which represents the Complex Number
data type.
Implement the following operations:
1. Constructor (including a default constructor which creates
the complex number 0+0i).
2. Overloaded operator+ to add two complex numbers.
3. Overloaded operator* to multiply two complex numbers.
4. Overloaded << and >> to print and read Complex Numbers.

*/


#include<iostream>
using namespace std;

class ComplexOp {

private:
      float img, real;

public:
      ComplexOp() {
            img = 0;
            real = 0;
      }
       friend ostream & operator << (ostream &out,  ComplexOp
&c);
       friend istream & operator >> (istream &in,  ComplexOp
&c);
      ComplexOp operator +(ComplexOp c2) {
            ComplexOp temp;

            temp.real = real + c2.real;
            temp.img = img + c2.img;

            return temp;
      }
      ComplexOp operator -(ComplexOp c2) {
            ComplexOp temp;

            temp.real = real - c2.real;
            temp.img = img - c2.img;

            return temp;
      }
      ComplexOp operator *(ComplexOp c2) {
            ComplexOp temp;
            temp.real=real*c2.real-(img*c2.img);
            temp.img=real*c2.img+(img*c2.real);
            return temp;
      }
```

```
};
istream & operator >> (istream &in,  ComplexOp &c)
{
    cout << "Enter Real Part ";
    in >> c.real;
    cout << "Enter Imaginary Part ";
    in >> c.img;
    return in;
}
ostream & operator << (ostream &out,  ComplexOp &c)
{

    out <<c.real<< "+"<<c.img<< "i" << endl;
    return out;
}
int main() {

      int ch;
      ComplexOp c1, c2, c3;
      cout<<"\tWelcome";
      do {
            cout << "1.Enter complex numbers.\n";
            cout << "2.Addition.\n";
            cout << "3.Subtraction.\n";
            cout << "4.Multiplication\n";
            cout << "5.Exit.\n";
            cout << "Enter your choice:\n";
            cin >> ch;
            switch (ch) {
            case 1:
                  cin >> c1;
                  cin >> c2;
                  cout<<"\n Complex numbers are: "<<c1<<","<<c2;
                  break;

            case 2:
                  c3 = c1 + c2;
                  cout<<"Complex numbers are: "<<c1<<","<<c2;
                  cout << "\n Addition:\n";
                  cout<<c3;
                  break;

            case 3:
                  c3 = c1 - c2;
                  cout<<"Complex numbers are: "<<c1<<","<<c2;
                  cout << "\n Subtraction:\n";
                  cout<<c3;
                  break;

            case 4:
                  c3 = c1 * c2;
                  cout<<"Complex numbers are: "<<c1<<","<<c2;
                  cout << "\n Multiplication:\n";
                  cout<<c3;
                  break;
            }
      } while (ch <5);
```

```
    return 0;
}
/*
OUTPUT:-
1.Enter complex numbers.
2.Addition.
3.Subtraction.
4.Multiplication
5.Exit.
Enter your choice:
1
Enter Real Part 8
Enter Imaginary Part 4
Enter Real Part 2
Enter Imaginary Part 3

 Complex numbers are: 8+4i
,2+3i
1.Enter complex numbers.
2.Addition.
3.Subtraction.
4.Multiplication
5.Exit.
Enter your choice:
2
Complex numbers are: 8+4i
,2+3i

 Addition:
10+7i
1.Enter complex numbers.
2.Addition.
3.Subtraction.
4.Multiplication
5.Exit.
Enter your choice:
3
Complex numbers are: 8+4i
,2+3i

 Subtraction:
6+1i
1.Enter complex numbers.
2.Addition.
3.Subtraction.
4.Multiplication
5.Exit.
Enter your choice:
4
Complex numbers are: 8+4i
,2+3i

 Multiplication:
4+32i
1.Enter complex numbers.
2.Addition.
3.Subtraction.
```

```
4.Multiplication
5.Exit.
Enter your choice:
5
*/
```

Experiment No 3.

Title: Calculator.

Aim: Write a C++ program to create a calculator for an arithmatic operator (+, -, *, /). The program should take 2 operands from user and perform the operation on those 2 operands depending upon the operator entered by user. Use a switch statement to select the operation. Finally, display the result.

Some sample instruction (interaction) with the program might look like this:-

```
Enter first number, operator, second number: 10 / 3
Answer = 3.333333.
Do another (y/n)? y
Enter first number, operator, second number: 12 + 100
Answer = 112.
Do another (y/n)? n
```

Prerequisites: Fundamentals of Programming language I and II

Objectives:-

- To understand the concept of classes and objects.
- To understand the concept of switch-case statement.

Theory:- 1) Class :-

Definition:- A class is a user defined data type. It is a collection of data members and member functions which act on those data members.

Classes are the base for object oriented programming, as objects are variables of type class.

Syntax:-

```
class class_name
{ ..... };
```

Example :-

```
class abc
{
  int a,b;
  public:
    int sum ();
};
```

2) Function:-

Definition :- A function is a named unit of a group of program statements. This unit can be invoked from other parts of the program as and when required.

In C++, a function must be defined before it is used anywhere in the program.
Syntax of function definition:-

```
type function_name(parameter_list)
{
//body of function
}
```

Switch-case statement:-
A switch-case statement is used where Multiple choices are needed. Nested if-else statements can also be used but it is better to use switch-case statement. While using switch-case statement, a variable which stores the choice is necessary.
Number of cases must be equal to number of choices.
How to use switch-case:-

```
cout << "In 1st Operand, Operator, 2nd Operand";
cin >> op;
switch (op)
  {
      case '/' : result = no1/no2;
       break;
      case '*': result = no1*no2;
       break;
```

```
case '+' : result = no1 + no2;
    break;
case '-' : result = no1 - no2;
    break;
default:
    cout << "invalid" << endl;
}
```

Now, here if user enters '/' as an operator the store in 'op' variable and then in switch it search the case for '/' and then execute "ca After each case "break" is necessary, otherwise i execute all further cases.

Faulities :- Linux Operating System, C(C, Eclipse framework

Algorithm :-

1. Start.
2. Create class calc having data members num num2, res and member functions add(), sub(), mul(), div(), getnos(), and display(
3. In main(), accept expression from user in format first no, operator and second no. Store the first no in a, operator in char c, a second no in int b.
4. Create object of class calc obj and call getn using obj.
5. In getnos(int, int), assign num1 = a and num2 = b

6. If c = '+', go to step 10.
7. if c = '-', go to step 11.
8. if c = '*' go to step 12.
9. if c = '/', go to step 13.
10. If c is a different operator, display error and go to step 15.
11. ~~if c is a different operator, display er~~
11. res = num1 + num2, go to step 15.
12. res = num1 - num2, go to step 15.
13. res = num1 * num2, go to step 15.
14. res = num1 / num2, go to step 15.
15. If user wishes to continue, go to step 3. Else go to step 16.
16. Stop.

Input: Accept 1st Operand, Operator and 2nd Operand.

Output: Result and Calculations.

Conclusion: Hence, we have studied concept of calculator using switch-case successfully.

```
//program to create an arithmetic calculator.
//Vrushil Soni-S1612037
#include<iostream>
using namespace std;
//begin class
class calculator
{
public:
    float num1,num2,result;
    void acc_input()
    {
        cout<<"ENTER THE NOS YOU WANT TO PERFORM OPERATIONS
ON:"<<endl;
        cin>>num1>>num2;
    }
    void display_result()
    {
        cout<<"THE RESULT OF THE OPERATION IS :"<<result
<<endl;
    }
    //function fo addition of num1 and num2
    void add_no()
    {
        result=num1+num2;
    }
    //function for subtraction of num1 and num2
    void sub_no1()
    {

            result=num2-num1;
    }
    void sub_no2()
    {
            result=num1-num2;
    }
    //multiplication funtion
    void mult_no()
    {
        result=num1*num2;
    }
    //function to divide nos
    void div_no1()
    {
        result=num1/num2;
    }
    void div_no2()
    {
        result=num2/num1;
    }
    void comp_num()
    {
        if(num1<num2)
            cout<<num2<<"IS GREATER THAN "<<num1<<endl;
        else
            cout<<num1<<"IS GREATER THAN "<<num2<<endl;
    }
```

```
};
//end of class calculator
int main()
{
    int ch,z,n,a;
    char ch1;
    calculator calc;
    z=0;
    do
    {
        cout<<"*************************************AVAILABLE
OPERATIONS*************************************"<<endl
        <<"1.INPUT NUMBERS\n2.ADD NUMBERS\n3.SUBSTRACT NUMBERS
\n4.MULTIPLY NUMBERS\n5.DIVIDE NUMBERS\n"<<endl;
        cin>>ch;
        switch(ch)
        {
            case 1:
                calc.acc_input();
                break;
            case 2:
                calc.add_no();
                calc.display_result();
                break;
            case 3:
                calc.comp_num();
                cout<<"FOR FIRST NUMBER SUBSTRACTED BY SECOND
NUMBER "<<endl;
                calc.sub_no1();
                calc.display_result();
                cout
<<"-----------------------------------------------------------
-----------------------------------"<<endl;
                cout<<"FOR SECOND NUMBER SUBSTRACTED BY FIRST
NUMBER "<<endl;
                calc.sub_no2();
                calc.display_result();
                break;
            case 4:
                calc.mult_no();
                calc.display_result();
                break;
            case 5:
                calc.comp_num();
                cout<<"FOR FIRST NUMBER DIVIDED BY SECOND
NUMBER "<<endl;
                calc.div_no1();
                calc.display_result();
                cout
<<"-----------------------------------------------------------
-----------------------------------"<<endl;
                cout<<"FOR SECOND NUMBER DIVIDED BY FIRST
NUMBER "<<endl;
                calc.div_no2();
                calc.display_result();
                break;
            default:
```

```
                cout<<"*****************************INVALID
CHOICE.PLEASE CHOOSE
AGAIN*********************************"<<endl;
        }
        cout<<"DO YOU WANT TO CONTINUE USING THE CALCULATOR?
\nY/N?"<<endl;
        cin>>ch1;
    }while(ch1=='y'||ch1=='Y');
    return 0;
}//end of main function.

//output

/************************************AVAILABLE
OPERATIONS*****************************************
1.INPUT NUMBERS
2.ADD NUMBERS
3.SUBSTRACT NUMBERS
4.MULTIPLY NUMBERS
5.DIVIDE NUMBERS

1
ENTER THE NOS YOU WANT TO PERFORM OPERATIONS ON:
20
55
DO YOU WANT TO CONTINUE USING THE CALCULATOR?
Y/N?
y
************************************AVAILABLE
OPERATIONS*****************************************
1.INPUT NUMBERS
2.ADD NUMBERS
3.SUBSTRACT NUMBERS
4.MULTIPLY NUMBERS
5.DIVIDE NUMBERS

2
THE RESULT OF THE OPERATION IS :75
DO YOU WANT TO CONTINUE USING THE CALCULATOR?
Y/N?
y
************************************AVAILABLE
OPERATIONS*****************************************
1.INPUT NUMBERS
2.ADD NUMBERS
3.SUBSTRACT NUMBERS
4.MULTIPLY NUMBERS
5.DIVIDE NUMBERS

3
55IS GREATER THAN 20
FOR FIRST NUMBER SUBSTRACTED BY SECOND NUMBER
THE RESULT OF THE OPERATION IS :35
-------------------------------------------------------------
---------------------------------
FOR SECOND NUMBER SUBSTRACTED BY FIRST NUMBER
THE RESULT OF THE OPERATION IS :-35
```

```
DO YOU WANT TO CONTINUE USING THE CALCULATOR?
Y/N?
y
**********************************AVAILABLE
OPERATIONS**************************************
1.INPUT NUMBERS
2.ADD NUMBERS
3.SUBSTRACT NUMBERS
4.MULTIPLY NUMBERS
5.DIVIDE NUMBERS

4
THE RESULT OF THE OPERATION IS :1100
DO YOU WANT TO CONTINUE USING THE CALCULATOR?
Y/N?
y
**********************************AVAILABLE
OPERATIONS**************************************
1.INPUT NUMBERS
2.ADD NUMBERS
3.SUBSTRACT NUMBERS
4.MULTIPLY NUMBERS
5.DIVIDE NUMBERS

5
55IS GREATER THAN 20
FOR FIRST NUMBER DIVIDED BY SECOND NUMBER
THE RESULT OF THE OPERATION IS :0.363636
------------------------------------------------------------
---------------------------------
FOR SECOND NUMBER DIVIDED BY FIRST NUMBER
THE RESULT OF THE OPERATION IS :2.75
DO YOU WANT TO CONTINUE USING THE CALCULATOR?
Y/N?
y
**********************************AVAILABLE
OPERATIONS**************************************
1.INPUT NUMBERS
2.ADD NUMBERS
3.SUBSTRACT NUMBERS
4.MULTIPLY NUMBERS
5.DIVIDE NUMBERS

6
**************************INVALID CHOICE.PLEASE CHOOSE
AGAIN*********************************
DO YOU WANT TO CONTINUE USING THE CALCULATOR?
Y/N?
n
*/
```

S1612037

Experiment No 4.

Title:- Student information system using ~~color~~ constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation.

Aim:- Develop an object oriented programming in C++ to create a database of student information system containing the following information. Name, roll-no, class, division, date of birth, blood group, contact, address, telephone number, driving license no, etc. Construct the database with suitable member functions for initializing and destroying the data viz constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators - new and delete.

Prerequisites: Fundamentals of Programming languages 1 and 2.

Objectives:- To learn the concept of constructor, default constructor, copy, destructor, static member functions, friend class, this pointer, inline code & dynamic memory allocation operators new and delete.

Theory:- A special method of the class that will be automatically invoked when an instance of the class is created is called as constructor. Following are the most useful features of constructors.

1) Constructor is used for initializing the values data members of the class.
2) Constructor is that whose name is same as class.
3) Constructor gets automatically called when an o class is created.
4) Constructors never have a return type even v
5) Constructor is of default, parameterized and Copy constructor.

The various types of constructor are as f Constructors can be classified into 3 types.

a] Default constructor.
b] Parameterised constructor.
c] Copy constructor.

A]. Default Constructor:- Default constructor is als as empty constructor which has no arguments is automatically called when we creates the of class but remembers name of constructor i same as name of class & constructor never with the help of void return type. If we never declare any arguments then #this is called as c constructor. Example:-

```
class abc
{
public:
abc() {...}
};
```

```
int main()
{ abc obj; // default constructor called.
  return 0;
}
```

B] Parameterized Constructor:- This is another type of constructor which has some arguments and same name as class name but it uses some arguments so for this we have to create object of class by passing some arguments at the time of creating object with the name of class. When we pass some arguments to the constructor then this will automatically pass the arguments to the constructor & the values will retrieve by the respective data members of the class. Example:-

```
class abc
{
  public:
  abc (int a, int b)
  {...}
};
void main()
{
  abc ob(5,4);
}
```

C] Copy Constructor:- This is also another type of constructor, In this constructor we pass the object of class into the another object of same class. As name suggests you copy, means Copy the values of one object

into the another object of class. This is used
copying the values of class object into an
object of class so we call them as copy co
& for copying the values we have to pass
name of object whose values we want to co
when we are using or passing an object to a
then we must have to use the & Ampersand
Address operator.

Destructor:-

As we know that constructor is that which
for assigning some values to data members & f
assigning some values this may also used so
memory so that to free up the memory whi
allocated by constructor, destructor is used whi
automatically called at the end of program an
don't have to explicitly call a destructor & de
can be parameterized or a copy. This can
only one means default ~~const~~ destructor which
no arguments. For declaring a destructor we h
use ntiled symbol in front of destructor.

Syntax:-

```
~class-name ()
{...}
```

Static Members:-

A class can contain static members, either
data or functions. A static variable has follo
properties:-

- It is initialized to zero when the first object of its class is created. No other initialization is permitted.
- Only one copy of that member is created for the entire class and is shared by all the objects of that class.
- It is ~~then~~ visible only within the class but its lifetime is entire program.

A static member function has following properties.

- A static function can have access to only other static members (fun or var) declared in the same class.
- A static function can be called using the class name instead of its object name.

Class_name :: fun_name;

Static member functions are considered to have class scope. In contrast to non static member functions, these functions have no implicit this argument; therefore, they can use only static data members, enumerators, or nested types directly. Static member functions can be accessed without using an object of the corresponding class type.

The following restrictions apply to such static functions

① They cannot access non static class member data using the member-selection operators (. or →)
② They cannot be declared as virtual.
③ They cannot have same name as a non static function that has the same argument types.

Friend Functions:-

In principle, private & protected members cannot be accessed from outside the same cla which they are declared. However, this rule affect friends. Friends are functions or classes as such. If we want to declare an external as friend of a class, thus allowing this functi access to the private and protected membe this class, we do it by declaring a prototype this external function within the class, and pr it with the keyword friend.

Properties of friend function:-

- It is not in the scope of the class to which it declared as friend.
- Since it is not in the scope of the class, it can called using the object of that class.
- It can be invoked like a normal function w/o help of any object.
- It can be declared in private or in public pa of the class.
- Unlike member functions, it cannot access the m names directly and has to use an object name dot operator with each member name.

Friend class:-

Just as we have the possibility to define friend function, we can also define a class as friend of another one, granting that second cl access to the protected and private member the first one.

'new' operator:-

Definition:- The new operator is used at the time of dynamic memory allocation and object construction.

Utilities:- Linux Operating Systems, GCC, Eclipse framework.

Algorithm:-

① Start.
② Read personal information such as name, date of birth, blood group, height, weight, insurance policy, number, contact address, telephone no., driving license no.
③ Print all information from database.
④ Stop.

Input:- Personal information such as Name, Date of Birth, blood group, height, weight, insurance policy, no., contact address, telephone no, driving license no.

Output:- Display Student information from database. The result in following format:-

Name DOB .... Driving License no.

1
2
3
.
.
.
.
.
.
.
n

Conclusion:- Hence, we have successfully studied concep
constructor, default constructor, copy constructor
destructor, static member functions, friend cl
pointer, inline code and dynamic memory allo
operators - new and delete.

```
/*NAME:- Vrushil Soni
CLASS:- S.E(C)
ROLL No:- S1612037
SUB:- OOP
                                        ASSIGNMENT NO    :- A_6
                                    Assignment NAME :- (Student info)

Develop an object oriented program in C++ to create a database
of student information
system containing the following information: Name, Roll
number, Class, division, Date of
Birth, Blood group, Contact address, telephone number, driving
licence no. etc Construct
the database with suitable member functions for initializing
and destroying the data viz
constructor, default constructor,Copy constructor, destructor,
static member functions,
friend class, this pointer, inline code and dynamic memory
allocation operators-new and delete.
*/
#include<iostream>
#include<string.h>
using namespace std;
class studentDb
{
      static int count;
      string name;
      int rollno;
      string dob;
      string dlic;
      string address;
      string contact;
      public:
            studentDb()
            {
                  name = "NAME";
                  rollno = 0;
                  dob = "00/00/0000";
                  dlic = "xxxxxxxxxx";
                  address = "ADDRESS";
                  contact = "0000000000";
                  count++;
            }
            studentDb(studentDb &ob)
            {
                  name = ob.name;
                  rollno = ob.rollno;
                  dob = ob.dob;
                  dlic = ob.dlic;
                  address = ob.address;
                  contact = ob.contact;
                  count++;
            }
            ~studentDb()
            {
                  cout<<"\n"<<this->name<<"Deleted";
            }
```

```
        inline void display()
        {
              cout<<"\n"<<nmae<<"\t"<<rollno<<"\t"<<dob
<<"\t"<<dlic
              <<"\t"<<address<<"\t"<<contact;
         }
         static void countObj()
         {
              cout<<"\n Object Count :"<<count;
         }
         friend class stud;
};
int studentDb::count=0;
class stud
{
public :
      void getData(studentDb &ob)
      {
            cout<<"\n Enter Name :";
            cin>>ob.name;
            cout<<"\n Enter Roll no :";
            cin<<ob.rollno;
            cout<<"\n Enter DOB :";
            cin<<ob.dob;
            cout<<"\n Enter Driving License No. :";
            cin>>ob.dlic;
            cout<<"\n Enter ADDRESS :";
            cin>>ob.address;
            cout<<"\n Enter Contact No. :";
            cin>>ob.contact;
       }
};

int main()
{
      studentDb s1;
      stud fs1;
      s1.display();
      fs1.getData(s1);
      s1.display();
      studentDb s2(s1);
      studentDb::countObj();
      s2.display();
      studentDb *ptr[5];
      studentDb::countObj();
      for(int i=0; i<5; i++)
      {
            ptr[i] = new studentDb();
            fs1.getData(*ptr[i]);
       }
       studentDb::countObj();
       for(int i=0; i<5; i++)
       {
            ptr[i]->display();
       }
       for(int i=0; i<5; i++)
       {
```

```
        delete(ptr[i]);
    }
}


/*
OUTPUT:

     Welcome to student database
Enter name of the student
XYZ
Enter Standard :
1
Enter Div :
A
Enter Roll no :
221016
Enter DOB :
22/02/1997
Enter Address :
FYNNBTCNTV
Enter Phone number :
9565522414
Enter License no :
6515724

*******Welcome***********

Name:ABC
Standard:2
Div:A
Roll no:221032 DOB:22/02/19976
Add:FYNNBTCNTV
Cell No:96666414
License:6515724

*/
```

S1612037.

Experiment No 5 -

Title :- Demonstrate class template for vector class.

Objectives:- 1) To learn and understand templates.
2) To demonstrate class template for vector class & its various operations.

Problem Statement:-
Create a class template to represent a generic vector. Include following member functions.

1) To create the vector.
2) To modify the value of a given element.
3) To multiply by a scalar value.
4) To display the vector in the form (10, 20, 30, ...).

Outcomes:- 1) Students will be able to learn and understand working and use of class template.
2) Students will be able to demonstrate class template for vector class with various operations.

Hardware requirements:-
Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software requirements:-
64 bit Linux / Windows Operating System, C++ compiler.

Theory:- Templates are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one. Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming & have been developed using template concept. There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector <int> or vector <string>.

Templates are a way of making your classes more abstract by letting you define the behaviours of the class. In essence, this is what is known as generic programming, this term is a useful way to think about templates because it helps remind the programmer that a templated class does not depend on the datatype (or types) it deals with. To a large degree, a templated class is more focused on the algorithms, though rather than the specific nuances of a single datatype. Templates can be used in conjunction with abstract datatypes in order to allow them to handle any type of data. For example, you could make a templated stack class that can handle a stack of any datatype, rather than having

to create a stack class for every different datatype for which you want to stack the function. The ability to have a single class that can handle several different datatypes means the code is easier to maintain, and it makes classes more reusable. The basic syntax for declaring a templated class is as follows:-

```
template <class a_type> class a_class {...};
```

The keyword 'class' above simply means that the identifier a_type will stand for a datatype. NB: a_type is not a keyword; it is an identifier that during the execution of the program will represent a single datatype. For example, you could, when defining variables in the class, use the following line:-

```
a_type a_var;
```

and when the programmer defines which datatype 'a_type' is to be when the program instantiates a particular instance of a_class, a_var will be of that type.

When defining a function as a member of a templated class, it is necessary to define it as a templated function:-

```
template <class a_type> void a_class<a_type>:: a_function()
{
    // body
}
```

When declaring an instance of a templated cl
Syntax is as follows:-
a-class <int> an-example-class;
An instantiated object of a templated class is c
specialization, the term specialization is use
remember because it reminds us that the or
class is a generic class, whereas a specific i
of a class is specialized for a single datatype
it is possible to template multiple types).

Usually when writing code it is easiest to
from concrete to abstract, therefore, it is easier
write a class for a specific datatype and then
to a templated-generic-class. For that bra
is the soul of wit, this example will be bori
therefore of little practical application.
We will define the first class to act only on
integers.

```
class calc
{
public:
  int multiply (int x, int y);
  int add (int x, int y);
};
int calc::multiply (int x, int y)
{
  return x*y;
}
int calc::add (int x, int y)
{ return x+y; }
```

We now have a perfectly harmless little class that functions perfectly well for integers; but what if we decided we wanted a generic class that would work equally well for floating point numbers? we would use a template.

```
template <class A_Type> class calc
{
  public:
  A_Type multiply (A_Type x, A_Type y);
  A_Type add (A_Type x, A_Type y);
};
template < class A_Type > A_Type calc<A_Type>::
                multiply (A_Type x, A_Type y)
{
  return x*y;
}
template <class A_Type> A_Type calc<A_Type>::
                              add (A_Type x, A_Type y)
{
  return x+y;
}
```

To understand the templated class, just think about replacing the identifier A_Type everywhere it appears, except as part of the template or class definition, with the keyword int. It would be the same as the above class; now when you instantiate an object of class calc you can choose which datatype the class will handle.

calc < double > a_calc_class;
Templates are handy for making your programs more generic and allowing your code to be reused lar

Vector:- A vector, in programming, is a type of arr that is one dimensional. Vector are a logical e in programming languages that are used for str data. Vectors are similar to arrays but their actual implementation and operation differs. Vectors are primarily used within the programming context of most programming languages and as data structure containers. Being a data str vectors are primarily used within the progra context of most programming languages and serve as data structure containers. Being a da structure, vectors are used for storing objects collections of objects in an organized structure. major difference between an array & a vector is unlike typical arrays, the containes size of a vector can be easily increased & decreased to complement different data storage types. Vectors have a dynamic structure and provide the ability to assign containes size up front and enable allocation of memory space quickly. Ve can be thought of as dynamic arrays.

Algorithm:-

1. Define a class vector of type template.
2. Declare one parameterized constructor and four methods to create vector, display vector, modify element in vector & multiply vector by a scalar value.
3. Define constructor outside the class with size passed as parameter.
4. Define method create to input size number of elements in vector.
5. Define display method to display size no of elements in vector.
6. Define vector modify method where index position of element to be modified is passed as argument.
7. Check if index position is not greater than size of vector
8. Accept new element from user & update element at given index position with new value.
9. Define multiply method with parameter as scalar value and multiply all elements in vector with given scalar value.
10. Define main method.
11. Accept size of vector from user
12. Accept data type of vector as either integer or float
13. Display menu of all available functions.
14. Accept choice of user.
15. If choice is 1 which must be 1st choice to input element in vector, call create method of vector class
16. for choice 2, enter index position to modify and call modify method
17. for choice 3, enter scalar value for multiplication & call multiply method:

(18) If choice is 4, display current elements in vector.
(19) For choice 5, exit the program.
(20) If user wants to continue, repeat steps 13 to 2
user enter 'n' or 5 as choice to exit prog
(21) End the program.

Test Cases:-

Test Case 1: (All valid inputs are given and desired output shown)

Enter size of vector : 5.

* * * MENU * * *
1. CREATE VECTOR
2. MODIFY VECTOR.
3. MULTIPLY VECTOR WITH SCALAR
4. DISPLAY VECTOR
5. EXIT.

Note: Create vector before doing any other operation.
Enter your choice.
1
Enter 5 elements in vector :-
2 5 3 12 55
Do you want to continue (y/n)? y
* * * MENU * * *
1. CREATE VECTOR
2. MODIFY VECTOR
3. MULTIPLY VECTOR WITH SCALAR.
4. DISPLAY VECTOR.
5. EXIT.
Note: Create vector before doing any other oper
Enter your choice.

4.
Elements in vector:-
2 5 3 12 55.
Do you want to continue (y/n) y
*** MENU ***
1. CREATE VECTOR.
2. MODIFY VECTOR.
3. MULTIPLY VECTOR WITH SCALAR.
4. DISPLAY VECTOR.
5. EXIT.
Note: Create vector before doing any other operation.
Enter your choice.
2
Enter index of element you want to modify: 3
Enter new value: 33
Do you want to continue (Y/N)? y
*** MENU ***
1. CREATE VECTOR
2. MODIFY VECTOR.
3. MULTIPLY VECTOR WITH SCALAR.
4. DISPLAY VECTOR.
5. EXIT.
Note: Create vector before doing any other operation.
Enter your choice.
4.
Elements in vector:-
2 5 33 12 55
Do you want to continue (Y/N)? y
*** MENU ***
1. CREATE VECTOR

2. MODIFY VECTOR.
3. MULTIPLY VECTOR WITH SCALAR.
4. DISPLAY VECTOR.
5. EXIT.

Note: Create vector before doing any other oper
Enter your choice.
3
Enter scalar value to multiply vector
2
Do you want to continue (y/n)? y

*** MENU ***
1. CREATE VECTOR.
2. MODIFY VECTOR.
3. MULTIPLY VECTOR WITH SCALAR
4. DISPLAY VECTOR.
5. EXIT.

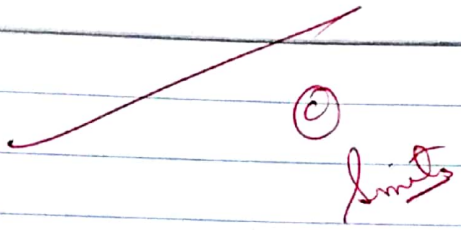
Note: Create vector before doing any other ope
Enter your choice:
4
Elements in vector:
4 10 66 24 110.
Do you want to continue (y/n)? n.

Test case 2: (If size is negative).
Enter size of vector :- 5 -
Size cant be negative. Initializing to 1.
Test Case 3: (If index of element to modify is greater than size)

Enter index of element you want to modify : 30
Index position can't be greater than size of vector.

Test case 4: (If index of element to modify is less than 0)
Enter index of element you want to modify: -3
Index cannot be negative - So initializing to 0.
~~for~~

sion:- Hence, we have demonstrated use of class template for vector class.



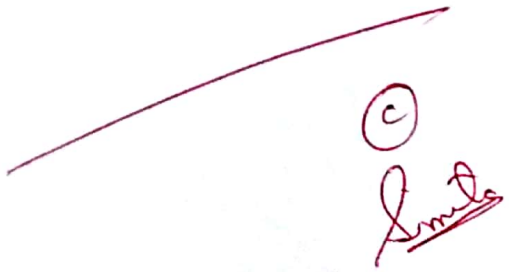
```
//program to create a generic vectorcand perform operations on
it
//Vrushil Soni-S1612037
#include<iostream>
using namespace std;
//syntax to create a generic class of type T
template<typename T>//typename or class is valid.
class Vector
{
public:
      T*vector;
       T size;
      void create_vector(T s)
      {
            size=s;
            vector=new int(size);//dynamically allocated size to
the vector
      }
      void set_elements(int i,T value)
      {
            vector[i]=value;//"i" is the iterator passed from
main and values are elements to be stored in vector
      }
      void display()
      {
            int i;
            cout<<"\n(";
            for(i=0;i<size;i++)
            {
                  cout<<vector[i]<<" ";

            }
            cout<<")";
      }
      void modify()
            {
            int i;
            cout<<"\n Enter position to modify:"<<endl;
            cin>>i;
            i--;//because index starts from 0
            cout<<"\n Enter value to modify:"<<endl;
            int new_val;
            cin>>new_val;
            vector[i]=new_val;//value modified
            cout<<"\n New modified values are :"<<endl;
            display();
      }
      void multiplay(int m)
      {
            int i;
            for(i=0;i<size;i++)
            {
                  vector[i]=vector[i]*m;
            }
            cout<<"\n New values are:"<<endl;
            display();
```

```
    }

};
//end of class


int main()
{
      int ch;
      char ch1;
      Vector<int> v;//class datatype is decided and object is
created.
      int i,size,value,m;
      do
      {

            cout
<<"*************************MENU*****************************
****"
            <<"\n \n1.Enter size of vectors:\n2.Create vector
\n3.Display\n4.Multiply vector by scalar value : \n5.Modify
some elements:\n"
            <<"\n Enter your choice: "<<endl;
            cin>>ch;
       switch(ch)
       {
                  case 1:
                        cout<<"\n Enter size of the vector:
 "<<endl;
                        cin>>size;
                        break;
                  case 2:
                        v.create_vector(size);
                        cout<<"\n Enter values:"<<endl;
                        for(i=0;i<size;i++)
                        {
                              cin>>value;
                              v.set_elements(i, value);
                        }
                        break;
                  case 3:
                        v.display();
                        break;
                  case 4:
                        cout<<"\n Enter the scalar value you want
 to multiply by : "<<endl;
                        cin>>m;
                        v.multiplay(m);
                        break;
                  case 5:
                        v.modify();
                        break;

                  default :
                        cout<<"!!!!!!!!!!!!!!!!!
```

```
INVALID!!!!!!!!!!!!!!!!!!"<<endl;
                    break;
            }
            cout<<"Do you want to continue?Y/N?"<<endl;
            cin>>ch1;
      }while(ch1='y'||ch1=='Y');

      return 0;

}//end of program

//output

/
**************************MENU*********************************
**

1.Enter size of vectors:
2.Create vector
3.Display
4.Multiply vector by scalar value :
5.Modify some elements:

 Enter your choice:
1

 Enter size of the vector:
3
Do you want to continue?Y/N?
y
**************************MENU*********************************
*

1.Enter size of vectors:
2.Create vector
3.Display
4.Multiply vector by scalar value :
5.Modify some elements:

 Enter your choice:
2

 Enter values:
2
4
6
Do you want to continue?Y/N?
y
**************************MENU*********************************
*

1.Enter size of vectors:
2.Create vector
3.Display
4.Multiply vector by scalar value :
5.Modify some elements:
```

```
 Enter your choice:
3

(2 4 6 )Do you want to continue?Y/N?
y
*****************************MENU*********************************
*

1.Enter size of vectors:
2.Create vector
3.Display
4.Multiply vector by scalar value :
5.Modify some elements:

 Enter your choice:
4

 Enter the scalar value you want to multiply by :
5

 New values are:

(10 20 30 )Do you want to continue?Y/N?
y
*****************************MENU*********************************
*

1.Enter size of vectors:
2.Create vector
3.Display
4.Multiply vector by scalar value :
5.Modify some elements:

 Enter your choice:
5

 Enter position to modify:
2

 Enter value to modify:
55

 New modified values are :

(10 55 30 )Do you want to continue?Y/N?
y
*****************************MENU*********************************
*

1.Enter size of vectors:
2.Create vector
3.Display
4.Multiply vector by scalar value :
5.Modify some elements:

 Enter your choice:
6
```

```
!!!!!!!!!!!!!!!!!INVALID!!!!!!!!!!!!!!!!!
Do you want to continue?Y/N?
n
*/
```



S1612037.

Experiment No. 6.

Title:- Book shop's inventory management using member functions and constructors.

Aim:- A book shop maintains the inventory of books that are being sold @ the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed, otherwise the message "Required copies not in stock" is displayed. Design a system using a class called books with suitable member functions and constructors. Use new operator in constructors to allocate memory space required. Implement C++ program for the system.

Prerequisites:- Fundamentals of Programming Language - I and II.

Objectives:-
- To study the representation, implementation and applications of data structures.
- To learn the concept of dynamic memory allocation & de-allocation (use of new & delete).

- To compare the benefits of static & dy
  data structures.

Theory:- The new & delete operators:-

There is following generic syntax to use operator to allocate memory dynamically
data-type
new data-type;

Here, data-type could be any built-in data including an array or any user defined data include class and structure. Let us start wi built-in data types. For example, we can de pointer to type double and then request the memory be allocated at execution time can do this using the new operator with following statements:-

```
double* pvalue = NULL; // Pointer initialized wit
pvalue = new double; // Request memory for
        variable.
```

The memory may not have been allocated if the free storage had been used up. So it good practice to check if new operator is NULL pointer and take- appropriate action below:-

```
double *pvalue = NULL;
if (!(pvalue = new double))
{
  cout << " Error: out of memory" << endl;
  exit (1);
}
```

Member Functions -

A member function of a class is a function that has its definition or its prototype within the class definition like any other variable. It operates on any object of the class of which it is a member, and has access to all the members of a class for that object.

```
class Box
{
  public:
    double length; // length of box.
    double breadth; // breadth of box.
    double height; // height of box.

    double getVolume (void)
    {
      return length * breadth * height;
    }
};
```

If you like you can define same function outside the class using scope resolution operator, as follows:-

```
double Box::getVolume(void)
{
   return length * breadth * height;
}
```

Let us take previously defined class to access members of the class using a member function instead of directly accessing them:

```
class Box
{
public:
double length; // Length of box.
double breadth; // breadth of box.
double height; // height of box.
double getVolume(void); // Returns box vo
};
```

Member functions can be defined within the class definition or separately using SRO(::) operator. Inside a member function within the class definition declares the function inline, even if you do not use the inline function specifier. So either you can define Volume() function as below:

class Box

Here, only important point is that you would have to use class just before :: operator. A member function will be called using a dot operator(.) on an object where it will manipulate data related to that object as follows :-

```
Box myBox; // Create an object.
myBox.getVolume(); // Call member fn of class by object
```

Facilities: Linux Operating System, GCC, Eclipse framework, VC++.

Algorithm:-

1. Start.
2. Read book details such as author, title, price, publishers and stock position.
3. Print all information of book.
4. Stop.

Input :- Details about books such as author, title, price, publisher, and stock position.

Output:- Display required book's details.

Conclusion:- Hence, we studied concept of new and delete operator successfully.

NAME:- Vrushil Soni
CLASS:- S.E(C)
ROLL No:- S1612037
SUB:- OOP

Assignment No:- A_12 (Bookshop)

A book shop maintains the inventory of books that are being sold at the shop.
The list includes details such as author, title, price, publisher and stock position. Whenever
a customer wants a book, the sales person inputs the title and author and the system searches
the list and displays whether it is available or not. If it is not, an appropriate message is displayed.
If it is, then the system displays the book details and requests for the number of copies required.
If the requested copies book details and requests for the number of copies required. If the requested
copies are available, the total cost of the requested copies is displayed; otherwise the message ?
Required copies not in stock? is displayed.Design a system using a class called books with suitable
member functions and Constructors. Use new operator in constructors to allocate memory space required.
Implement C++ program for the system.

*/

```
#include<iostream>
#include<string.h>
using namespace std;
class bookshop
{
public:
      char title[20];
      char author[20];
      char publisher[20];
      float price;
      int no_of_copies,edition;
      void insertinfo();
      void acceptinfo();
      int searchinfo(char title[],char author[]);
      void num_of_copies(int);
      bookshop()
      {
            char *title=new char[20];
            char *author=new char[20];
            char *publisher=new char[20];
            price=0.0;
            edition=0;
            no_of_copies=0;
      }
};
void bookshop::insertinfo()
{
      cout<<"\n\n Enter the title: ";
      cin>>title;
```

```
    cout<<"Enter the author: ";
    cin>>author;
    cout<<"Enter publisher name: ";
    cin>>publisher;
    cout<<"Enter price: ";
    cin>>price;
    cout<<"Enter num_of_copies: ";
    cin>>no_of_copies;
    cout<<"Enter edition: ";
    cin>>edition;
}
void bookshop::acceptinfo()
{
    cout<<"\n title:"<<title;
    cout<<"\n author:"<<author;
    cout<<"\n publisher:"<<publisher;
    cout<<"\n price:"<<publisher;
    cout<<"\n no_of_copies:"<<no_of_copies;
    cout<<"\n edition:"<<edition;
}
int bookshop::searchinfo(char t[],char a[])
{
    {
        if(strcmp(t,title)&&strcmp(a,author))
        {
            return 0;
        }
        else
        {
            return 1;
        }
    }
}
void bookshop::num_of_copies(int num)
{
    if(no_of_copies>=num)
    {
        cout<<"\n Cost of "<<num<<" books is Rs."<<(price
*num);
    }
    else
    {
        cout<<"\n sorry! These many copies are not
available";
    }
}
int main()
{
    int i,n;
    int flag=0;
    char t[20],a[20];
    cout<<"\n *** Details of books are as follow: ***";
    bookshop b[20];
    for(i=0;i<2;i++)
    {
        b[i].insertinfo();
```

```
        b[i].acceptinfo();
    }

    cout<<"\n\n Enter title to search:";
    cin>>t;
    cout<<"\n Enter author to search:";
    cin>>a;
    for(i=0;i<2;i++)
    {
          if(1==b[i].searchinfo(t,a))
          {
                flag=1;
                      break;
          }
      }
      if(flag==0)
      {
           cout<<"\n Book is  not found";
      }
      else
      {
           cout<<"\n Book is found";
      }
      cout<<"\n How many copies?";
      cin>>n;
      b[i].num_of_copies(n);
      return 0;
}


      /* OUTPUT

 *** Details of books are as follow: ***

 Enter the title: OOP
Enter the author: J_s_katre
Enter publisher name: techmax
Enter price: 600
Enter num_of_copies: 3
Enter edition: 2016

 title:OOP
 author:J_s_katre
 publisher:techmax
 price:techmax
 no_of_copies:3
 edition:2016

  Enter the title: DSA
 Enter the author: katre
 Enter publisher name: Technical
 Enter price: 700
 Enter num_of_copies: 4
 Enter edition: 2016

  title:DSA
  author:katre
```

```
publisher:Technical
price:Technical
no_of_copies:4
edition:2016

Enter title to search:OOP

Enter author to search:J_s_katre

Book is found
How many copies?2

Cost of 2 books is Rs.1200 */
```

S1612037.

Experiment No. 9.7.

Title:- Employee bio-data using inheritance.

Aim:- Create employee bio-data using following classes- ① Personal record. ② Professional record. ③ Academic record. Assume appropriate data members & member function to accept required data & print bio-data. Create bio-data using multiple inheritance using C++/Java/Python.

Prerequisites:- Fundamentals of Programming Language- I & II.

Objectives:- To learn concept of Inheritance.

Theory:-

Inheritance -

Inheritance is the process by which objects acquire the properties of objects of other class. In OOP, inheritance provides reusability, like, adding additional features to an existing class without modifying it. This is achieved by deriving a new class from the existing one. The new class will have combined features of both the classes.

Types of Inheritance:-

① Single inheritance.
② Multiple inheritance.
③ Multilevel inheritance.
④ Hierarchical inheritance.
⑤ Hybrid inheritance.

There are 3 types of class inheritance : public, p and protected.

The protected access specifier is similar to priva only difference occurs in fact with inheritan a class inherits from another one, the memb the derived class can access the protected members inherited from the base class, but its private members.

Difference between public, private & protected.

- A member (either data member or member f declared in a private section of a class can only accessed by member functions and friends of the class.
- A member (either data member or member fun declared in a protected section of a class can be accessed by member functions and friends of that class, and by member functions & friend functions of derived classes.
- A member (either data member or member fun declared in a public section of a class can be accessed by anyone.

The following table indicates how the attribut are inherited in 3 different types of inherita

Access specifier in the base class:-

	Private.	Public	Protected.
public inheritance.	The member is private.	The member is public.	The member is protected.
Private inheritance.	The member is private.	The member is private.	The member is private.
Protected inheritance.	The member is private.	The member is protected.	The member is protected.

What a derived class inherits:-

- Every data member defined in the parent class (although such members may not always be accessible in derived class!).
- Every ordinary member function of the parent class (although such members may not be always accessible in derived class!).
- The same initial layout as the base class.

What a derived class doesn't inherit.

- The base class's constructors and destructors.
- The base class's assignment operator.
- The base class's friends.

What a derived class can add-

- New data members.
- New member functions.
- New constructors and destructors.
- New friends.

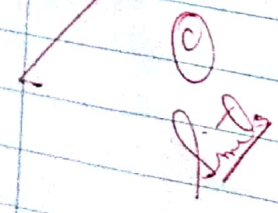
Facilities:- Linux Operating Systems, GCC, Eclipse Framework

Algorithm:
1. Start.
2. Read employee information i.e. personal, professional and academic.
3. Print all information from database.
4. Stop.

Input:- Base class which consist data members name employee, qualification and etc.
Three derived classes (B1, B2 and B3) which contain data member relevant to personal, professional and academic details.

Output:- 1. Display table containing all data members.
2. Insert a new entry.

Conclusion:- Hence, we have successfully studied concept of inheritance.



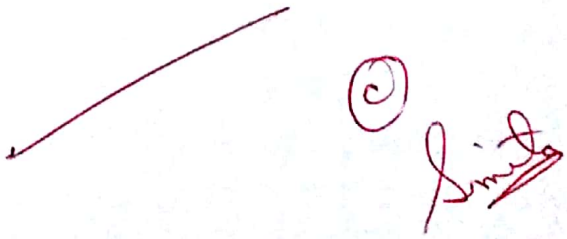
```
//program tto implement multiple inheritance
//Vrushil Soni-S1612037
#include<iostream>
#include<string.h>
#define MAX 100

using namespace std;
//class with personal data.
class personal
{
      public:
                        string name,address;
                        long int phno;
                        string dob;
      public:
                         void displayp()
                        {
                              cout<<"\nName - "<<name;
                              cout<<"\nAddress - "<<address;
                              cout<<"\nPhone no. - "<<phno;
                              cout<<"\nDate of birth - "<<dob;
                        }
};

//class with academic data

class academics
{
      public:
              string quali;
             float ssc,hsc;

      public:
           void displaya()
                 {
                       cout<<"\nQualification: "<<quali;
                       cout<<"\nSSC percentage: "<<ssc<<"%";
                       cout<<"\nHSC percentage: "<<hsc
<<"%"<<endl;
                 }

};

//class with professional data
class professional
{
      public:
                        int exp;
                        string company,work;

      public:
                              void displaypr()
                              {
                                    cout<<"\nProfessional
experience: "<<exp<<" yrs";
                                    cout<<"\nPresently working in
company: "<<company;
                                    cout<<"\nCurrent designation:
```

```
"<<work<<endl;
                    }
};
//class biodata that will inherit all the other classes
class biodata:public personal,public academics,public
professional
{
    public:
        void accept();
        void display()
        {
            personal::displayp();
            academics::displaya();
            professional::displaypr();
        }
};

void biodata::accept()
{
    cout<<"Enter Name: \n";
    getline(cin,name);
    getline(cin,name);

    cout<<"Enter Address: \n";
    getline(cin,address);

    cout<<"Enter date of birth(DD/MM/YYYY):\n";
    getline(cin,dob);

    cout<<"Enter phone no.: \n";
    cin>>phno;

    cout<<"Enter experience achieved(years): \n";
    cin>>exp;

    cout<<"Enter company name: \n";
    getline(cin,company);
    getline(cin,company);

    cout<<"Enter current designation in "<<company<<endl;
    getline(cin,work);

    cout<<"Enter qualification: \n";
    getline(cin,quali);

    cout<<"Enter SSC percentage(Out of 100):\n";
    cin>>ssc;

    cout<<"Enter HSC percentage(Out of 100): \n";
    cin>>hsc;
}//end of class biodata.

int main()
{
    biodata b[MAX];
```

```
    int empno;
    int z,ch;
    char ch1;
    z=0;
    do
    {
          cout<<"1.Create database\n2.Display
database."<<endl;
          cin>>ch;
          switch(ch)
          {
               case 1:
                    cout<<"Enter the no of employees you want
to store information about: "<<endl;
                    cin>>empno;
                    for(int i=0;i<empno;i++)
                    {
                         cout
<<"_____"<<endl;
                              b[i].accept();
                    }
                    break;
               case 2:
                    for(int i=0;i<empno;i++)
                    {
                         cout
<<"_____"<<endl;
                              b[i].display();
                    }
                    break;
          }
          cout<<"Do you want to contiue?Y/N?"<<endl;
          cin>>ch1;
     }while(ch1=='y'||ch1=='Y');
     return 0;
}//end of program

//output

/*1.Create database
2.Display database.
1
Enter the no of employees you want to store information about:
1
_____
Enter Name:
Vrushil Soni
Enter Address:
Balaji Nagar
Enter date of birth(DD/MM/YYYY):
23/03/1999
Enter phone no.:
7875617510
Enter experience achieved(years):
5
Enter company name:
Google
```

```
Enter current designation in Google
CEO
Enter qualification:
M.Tech
Enter SSC percentage(Out of 100):
88
Enter HSC percentage(Out of 100):
70
Do you want to contiue?Y/N?
y
1.Create database
2.Display database.
2
```

```
Name - Vrushil Soni
Address - Balaji Nagar
Phone no. - 7875617510
Date of birth - 23/03/1999
Qualification: M.Tech
SSC percentage: 88%
HSC percentage: 70%

Professional experience: 5 yrs
Presently working in company: Google
Current designation: CEO
Do you want to contiue?Y/N?
n*/
```



Experiment NO : 8

Title: To demonstrate file handling operations to create phonebook.

Objectives:
1) To learn and understand file handling functions.
2) To demonstrate random accessing file.

Problem statement: Write a menu driven program that will create a data file containing the list of telephone numbers in the following form John 23456 Ahmed 9876 .....
Use a class object to store each set of data, access the file created & implement the following tasks.
1. Determine the telephone no. of specified person.
2. Determine the name if telephone no is known.
3. Update the telephone no. whenever there is a change.

Outcomes:
1) Students will be able to learn & understand file handling operations.
2) Students will be able to demonstrate random accessing file.

Hardware requirements: Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software requirements: 64 bit Linux / Windows Operating system / , G++ compiler.

Theory:- Opening a file:- A file must be opened before it can read from it or write it. Either the ofstream fstream object may be used to open a file for writing and ifstream object is used to open file for reading purpose only.
Following is the standard syntax for open() function which is a member of fstream, ifstream and ofstream objects.

void open(const char *filename, ios::openmode mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.

Closing a file:-

When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for close() function which is a member of fstream, ifstream and ofstream objects.

void close();

Writing to a file:-

While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as

use that operator to output information to the screen. The only difference is that you use an ofstream or fstream object instead of cout object.

Reading from a file:-

You read information from a file into your program, using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you can use an ifstream or fstream object instead of the cin object.

File Pointers and Manipulators:-

- Each file has 2 pointers known as file pointers, one is called the input pointer and the other is called output pointer.
- The input pointer is used for reading the contents of a given file location and the output pointer is used for writing to a given file location.
- Each time an input or output operation takes place, the appropriate pointer is automatically advanced.

Functions for manipulations of file pointer:-

- All the actions on the file pointers take place by default.
- For controlling the movement of file pointers file stream classes support the following functions:
- seekg() moves get pointer (input) to a specified loc.
- seekp() moves put pointer (output) to a specified loc.

- tellg() Give the current position of get pointer
- tellp() Give the current position of put pointer
- For example, the statement seekg(10); moves the file pointer to the byte number 10. The bytes in a file are numbered beginning from zero.
- Therefore, the pointer to the 11th byte in the file.

Consider the following statements -

```
ofstream fileout;
fileout.open("hello", ios::app);
int p = fileout.tellp();
```

- On execution of these statements, the output pointer is moved to the end of the file 'hello' and the value of p will represent the number of bytes in the file.

Specifying the Offset:-

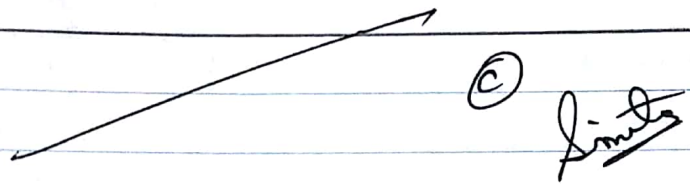
- 'seek' functions seekg() & seekp() can also be used with 2 arguments as follows:-

```
seekg(offset, refposition);
seekp(offset, refposition);
```

- The parameter offset represents the number of bytes the file pointer is to be moved from the location specified by parameter refposition.
- The refposition takes one of the following constants defined in the ios class.
- ios::beg start of file.
- ios::cur current position of pointer.
- ios::end end of file.

- The seekg() function moves the associated file's 'get' pointer while the seekp() function moves the associated file's 'put' pointer.

Conclusion:- Hence, we have studied and demonstrated file operations and randomly accessing file for performing various file operations.



```
//program to create a telephone directory and perform
operations
//Vrushil Soni-S1612037
//program for file input and output.
#include<iostream>
#include<string.h>
#include<fstream>
#define MAX 100
using namespace std;
struct data
{
      string name;
      long int tele;
};
class directoryio
{
      public:
            long int flag,temp_no;
            string temp_name;
            struct data d;
            //functions to create database for directory and
store into a text file .
            void inpdata()
            {
                  getline(cin,d.name);
                  getline(cin,d.name);
                  cin>>d.tele;
            }
            void createdir()
            {
                  ofstream tel;
                  tel.open("/home/vrushil/Desktop/TELEPHONE
DIRECTORY.txt",ios::in|ios::app|ios::ate);//file will work for
input and append.not sure about append and ate.
                  tel<<d.name<<"\t"<<d.tele<<endl;;
                  tel.close();
            }
            void writedir()
            {
                  ifstream telip;
                  telip.open("/home/vrushil/Desktop/TELEPHONE
DIRECTORY.txt");
                  telip.close();
            }
            //display function
            void dispdata()
            {
                  cout<<d.name<<"\t"<<d.tele<<endl;
            }
            //search by name
            void search_dir(string sname)
            //no result if entry not found.
                  {
                      if(d.name==sname)
                      {
                          cout<<"!!!!!!!!!!!
FOUND!!!!!!!!!!!!!!!!!!!!!!"<<endl
```

```
                    <<d.name<<"\t"<<d.tele<<endl;
                flag=1;
                    }

                }
                //search by number
                void search_dirbyno(long int sno)
            //no result if entry not found.
                {
                    if(d.tele==sno)
                    {
                        cout<<"!!!!!!!!!!!
FOUND!!!!!!!!!!!!!!!!!!!!"<<endl
                        <<d.name<<"\t"<<"\t"<<d.tele<<endl;
                flag=1;
                    }
                }

                ///////function to update or modify data.
            void update_dir(long int new_number,string new_name)
                {
                //swapping new name with old one
                {
                      d.name=temp_name;
                      d.name=new_name;
                      temp_name=new_name;
                }
                //swapping new number with old one
                {
                      d.tele=temp_no;
                      d.tele=new_number;
                      temp_no=new_number;
                }
                cout<<"UPDATED DATA IS:\n"<<endl;//call for
display function.

                }



};//end of class directoryio
int main()
{
     ofstream tel;
     tel.open("/home/vrushil/Desktop/TELEPHONE
DIRECTORY.txt",ios::trunc);//to delete the old data from the
TXT file.
     tel.close();
       long int i,z,n,sno,new_number;
     char ch;
       string sname,new_name;

     cout<<"ENTER THE NO. OF INPUTS  IN THE DIRECTORY"<<endl;
     cin>>i;
     directoryio dir[MAX];
     z=0;
```

d. name = new_name

```
    do
    {
          cout
<<"*****************************************MENU**********************
*****************"<<endl;
          cout<<"1.CREATE DATABASE\n2.DISPLAY DATABASE
\n3.SEARCH AND UPDATE DATABASE\n"<<endl;
          cin>>n;
          switch(n)
          {
               case 1:
                    cout<<"CREATE DIRECTORY"<<endl
<<"*********************************************"<<endl
                         <<"ENTER NAME AND NUMBER IN THE
DIRECTORY"<<endl;
                    for(int j=0;j<i;j++)
                    {
<<"                         cout
                                                          "<<endl;
                              dir[j].inpdata();
                               dir[j].createdir();
                         dir[j].writedir();
                    }
                    break;
               case 2:
                    cout
<<"**************DIRECTORY****************"<<endl;
                    cout<<"NAME\t\tCONTACT NO."<<endl;
                    for(int j=0;j<i;j++)
                    dir[j].dispdata();
                    break;
                      case 3:
                          int chs,v;
                          cout<<"1.SEARCH BY NAME\n2.SEARCH
BY NUMBER\n"<<endl;
                          cin>>chs;
                          v=0;
                          switch(chs)
                          {
                              case 1:
                          cout<<"ENTER THE NAME YOU WANT TO
SEARCH"<<endl;
                                   getline(cin,sname);
                                   getline(cin,sname);
                                   for(int j=0;j<i;j++)
                                   {
                                        dir[j].search_dir
(sname);
                          if(dir[j].flag==1)
                          {
                               char chup;
                               cout<<"DO YOU WANT TO UPDATE
THIS ENTRY?Y/N?"<<endl;
                               cin>>chup;
                               if(chup=='y'||chup=='Y')
                               {
                                    }
```

```
                                        cout<<"ENTER THE NEW
NAME AND NUMBER!"<<endl;
                                        getline
(cin,new_name);//find why to getlines are needed to execute
properly,
                                        getline(cin,new_name);
                                        cin>>new_number;
                                        dir[j].update_dir
(new_number,new_name);
                                        dir[j].createdir
();//might be a problem
                                        dir[j].dispdata();
                                    }
                                    else
                                        cout<<"!!!!!!!!!!!
THANKYOU!!!!!!!!!!!!!!!"<<endl;
                                    break;
                                }
                                else
                                    cout
<<".............SEARCHING............."<<endl;
                                    }
                                    break;
                                case 2:
                                cout<<"ENTER THE NUMBER YOU WANT TO
SEARCH"<<endl;
                                    cin>>sno;
                                    for(int j=0;j<i;j++)
                                    {
                                        dir[j].search_dirbyno
(sno);
                                if(dir[j].flag==1)
                                {
                                    char chup;
                                    cout<<"DO YOU WANT TO UPDATE
THIS ENTRY?Y/N?"<<endl;
                                    cin>>chup;
                                    if(chup=='y'||chup=='Y')
                                    {
                                        cout<<"ENTER THE NEW
NAME AND NUMBER!"<<endl;
                                        getline
(cin,new_name);//goes into infinite loop if only one getline
is used!!find solution
                                        getline(cin,new_name);
                                        cin>>new_number;
                                        dir[j].update_dir
(new_number,new_name);
                                        dir[j].createdir
();//ipdated data will remain with the old data in the txt
```

```
file.                                   //might be a problem.
                                        dir[j].dispdata();
                                    }
                                    else
                                        cout<<"!!!!!!!!!!!
THANKYOU!!!!!!!!!!!!!!!"<<endl;
                                    break;
                                }


                                else
                                    cout
<<".........SEARCHING............"<<endl;

                                        }

                                        break;
                                    default:
                                        cout
<<"!!!!!!!!!!!!!!!!!!!!!!INVALID!!!!!!!!!!!!!!!!!!!!!!"<<endl;
                                        break;
                                }
                                break;

                                default:
                            cout<<"!!!!!!!!!!!!!!!INVALID
CHOICE!!!!!!!!!!!!!!!!"<<endl;
                                    break;
            }
            cout<<"**************************************"<<endl
<<"DO YOU WANT TO CONTINUE USING THIS DIRECTORY?(Y/N)"<<endl;
            cin>>ch;
      }while(ch=='y'||ch=='Y');

      return 0;
}//end of program.

//output

/*ENTER THE NO. OF INPUTS  IN THE DIRECTORY
5
**********************************MENU**********************
**************
1.CREATE DATABASE
2.DISPLAY DATABASE
3.SEARCH AND UPDATE DATABASE

1
CREATE DIRECTORY
**************************************
ENTER NAME AND NUMBER IN THE DIRECTORY
_____________________________________________
Vrushil
7875617510
_____________________________________________
Bhargav
```

```
9623444333
NIKHIL
7038948969
Aishwarya
7057509732
Tanvi
7798038458
****************************************
DO YOU WANT TO CONTINUE USING THIS DIRECTORY?(Y/N)
Y
****************************************MENU**********************
**************
1.CREATE DATABASE
2.DISPLAY DATABASE
3.SEARCH AND UPDATE DATABASE

2
**************DIRECTORY***************
NAME        CONTACT NO.
Vrushil 7875617510
Bhargav 9623444333
NIKHIL      7038948969
Aishwarya 7057509732
Tanvi    7798038458
****************************************
DO YOU WANT TO CONTINUE USING THIS DIRECTORY?(Y/N)
Y
****************************************MENU**********************
**************
1.CREATE DATABASE
2.DISPLAY DATABASE
3.SEARCH AND UPDATE DATABASE

3
1.SEARCH BY NAME
2.SEARCH BY NUMBER

1
ENTER THE NAME YOU WANT TO SEARCH
Tanvi
............SEARCHING............
............SEARCHING............
............SEARCHING............
............SEARCHING............
!!!!!!!!!!!FOUND!!!!!!!!!!!!!!!!!!!
Tanvi 7798038458
DO YOU WANT TO UPDATE THIS ENTRY?Y/N?
Y
ENTER THE NEW NAME AND NUMBER!
Abhishek
7038304627
UPDATED DATA IS:

Abhishek 7038304627
```

```
****************************************
DO YOU WANT TO CONTINUE USING THIS DIRECTORY?(Y/N)
Y
****************************************MENU**************************
***************
1.CREATE DATABASE
2.DISPLAY DATABASE
3.SEARCH AND UPDATE DATABASE

2
***************DIRECTORY*****************
NAME        CONTACT NO.
Vrushil 7875617510
Bhargav 9623444333
NIKHIL      7038948969
Aishwarya 7057509732
Abhishek 7038304627
****************************************
DO YOU WANT TO CONTINUE USING THIS DIRECTORY?(Y/N)
N*/
```

Experiment No: 9

Title: Exception Handling.

Aim: Create user defined exception to check the following conditions and throw exception if the criteria does not met.

a) User has age between 18 and 55.
b) User stays has income between Rs 50K to 100K per month.
c) User stays in Pune / Mumbai / Bangalore / Chennai.
d) User has 4-wheeler.

Accept age, Income, City, Vehicle from the user & check for the conditions mentioned above. If any of the condition not met then throw exception.

Prerequisites: FPL I and II.

Objectives:
- To understand the concept of Exception Handling.

Theory:

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from 1 part of a program to another.

C++ exception handling is built upon 3 keywords: try, catch and throw.

- throw: A program throws an exception when a problem shows up. This is done using a throw keyword.
- catch: A program throws an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.
- try: A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch block looks like the following:-

```
try
{
  // protected code
} catch ( ExceptionName e1)
  {
    // catch block
  }

catch ( ExceptionName e2)
{
  // catch block
} catch ( ExceptionName eN)
  { // catch block
  }
```

Throwing Exceptions:-
Exceptions can be thrown anywhere within a code block using throw statements. The operand of the throw statements determines a type for the exception and can be any expression and the type of the result of expression determines the type of exception thrown.
Following is an example of throwing an exception when dividing by zero condition occurs:-

```
double division( int a, int b)
{
   if (b == 0)
     {
       throw "Division by zero condition!";
     }
   return (a/b);
}
```

Catching exceptions:-
The catch block following the try block catches any exception. You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parenthesis following the keyword catch.

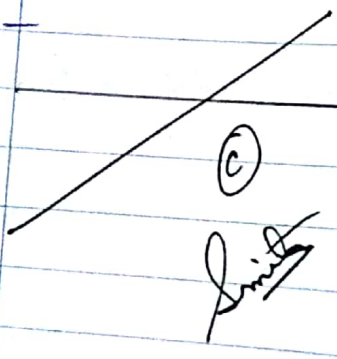
```
try
{
  // protected code.
} catch ( ExceptionName e)
  { // code to handle ExceptionName exception)
    }
```

Facilities: Linux Operating Systems, GCC, Eclipse framework, V

Algorithm:- Step 1: Start.
Step 2: Read television details such model num
screen size in inches and the price.
Step 3: Print data members.
Step 4: Stop.

Input: Accept age, income, city, vehicle.
Output:- If exception is caught display error message.

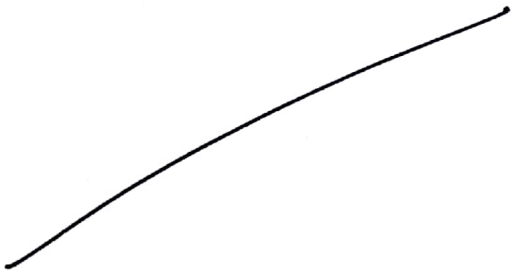
Conclusion: Hence, we have successfully studied concept of
exception handling.



```
//program to implement exception handling in c++
//Vrushil Soni-S1612037
#include <iostream>
#include <string.h>
using namespace std;
class except_data
{
public:
      int age;
      string city,name;
      long income;
      char fourwheeler;
      int count;
public:
      void accept_age()
      {
            cout<<"Enter your name : "<<endl;
            getline(cin,name);
            try
            {
                  cout<<"Enter your age:"<<endl;
                  cin>>age;
                  if(age<18||age>55)
                        throw(age);
                  else
                        count++;
            }
            catch(int i)
            {
                  cout<<name<<" Your age should be between 18-55
years"<<endl;
            }
      }
      void accept_city()
      {
            try
            {
                  cout<<"Enter the city you live in:"<<endl;
                  cin>>city;
                  if
(city=="Pune"||city=="PUNE"||city=="pune"||city=="MUMBAI"||cit
y=="Mumbai"||city=="mumbai"||city=="banglore"||city=="BANGLORE
"||city=="Bangalore"|city=="CHENNAI"||city=="chennai"||city==""
Chennai")
                        count++;
                  else
                        throw(city);
            }
            catch(string s)
            {
                  cout<<name<<" Your City should be either Pune,
Mumbai, Bangalore or Chennai"<<endl;
            }
      }
      void accept_income()
      {
            try
```

```
        {
                cout<<"Enter monthly income:"<<endl;
                cin>>income;
                if(income<50000||income>100000)
                        throw(income);
                else
                        count++;
        }
        catch(double d)
        {
                cout<<name<<" income should be between Rs.
50,000- Rs. 1,00,000"<<endl;
        }
    }
    void accept_fourw()
    {
        try
        {
                cout<<"Do you have a four wheeler? Y or
N"<<endl;
                cin>>fourwheeler;
                if(fourwheeler=='y'||fourwheeler=='Y')
                        count++;
                else
                        throw(fourwheeler);
        }
        catch(char fw)
        {
                cout<<name<< " four wheeler not
present"<<endl;
        }
    }
    void display_go()
    {
        if(count==4)
                cout<<"No exceptions were found in provided
data!!";
    }
};
int main()
{
    except_data obj;
    obj.accept_age();
    obj.accept_income();
    obj.accept_city();
    obj.accept_fourw();
    obj.display_go();
    return 0;
}//end of program

//output

/*Enter your name :
Vrushil Soni
Enter your age:
18
Enter monthly income:
```

```
50000
Enter the city you live in:
Pune
Do you have a four wheeler? Y or N
n
Vrushil Soni four wheeler not present

output 2

Enter your name :
Vrushil Soni
Enter your age:
18
Enter monthly income:
70000
Enter the city you live in:
Pune
Do you have a four wheeler? Y or N
y
No exceptions were found in provided data!!
*/
```



Experiment No: 10

Title: Demonstrate various file operations using C++.

Objectives:-
1) To learn & understand streams & files in object oriented paradigm.
2) To demonstrate file operations like create, open, read, write and close a file.

Problem Statement: Write a C++ program that creates an output file, writes information to it, closes the file and open it again as an input file & read the information from the file.

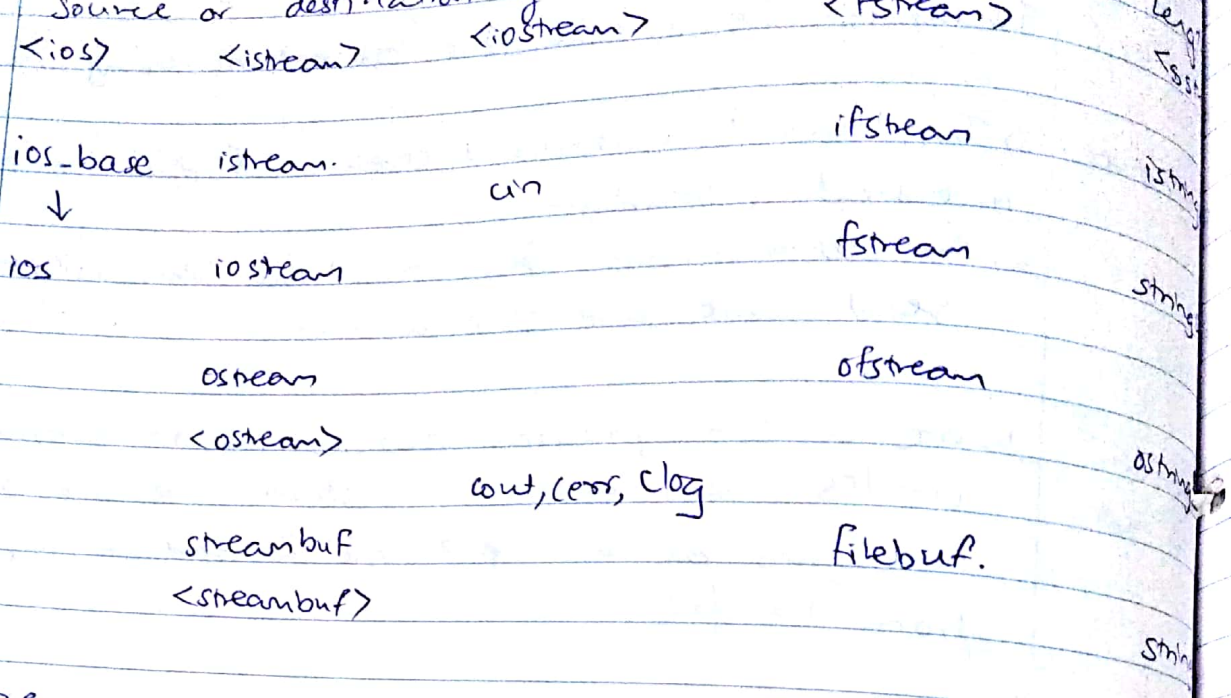
Outcomes:-
1) Students will be able to learn & understand concepts of stream and files in C++.
2) Students will be able to demonstrate various operations like creating a file, opening an existing file, reading from file, writing in file, closing file.

Hardware Requirements: Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements: 64 bit Linux / Windows Operating System, C++ compiler.

Theory:- The iostream library is an object-oriented library that provides input and output functionality using streams. A stream is an abstraction that represents

a device on which input & output operations can be performed. A stream can basically be represented as a source or destination of characters of indefinite length



Streams are generally associated to a physical source or destination of characters, like a disk file, the keyboard or the console, so the characters gotten or written to/from our abstraction called stream are physically input/output to the physical device. For example, file streams are C++ objects to manipulate and interact with files; once a file stream is used to open a file, any input or output operation performed on that stream is physically reflected in the file.

So far, we have been using the iostream standard library which provides cin and cout methods for reading from standard input & writing to standard output respectively. This tutorial will teach you how to read and write from a file. This requires another standard C++ library called fstream, which defines three new data types:-

Data type	Description
ofstream.	This data type represents the output file stream and is used to create file and to write information to files.
ifstream.	This data type represents the input file stream and is used to read information from files.
fstream.	This data type represents the file stream generally, and has the capabilities of both ifstream and ifstream which means it can create files, write information to files, and read information from files.

To perform file processing in C++, header files <iostream> & <fstream> must be included in your C++ source file. These classes are derived directly or indirectly from the classes istream and ostream. We have already used objects whose types were these classes: cin is an object of class istream and cout is an object of class ostream. Therefore, we have already been using classes that are related to our file streams. And in fact, we can use our file streams the same way we are already used to use cin and cout with the only difference that we have to associate these streams with physical files. Let's see an example:-

```
// basic file operations:
#include <iostream>
#include <fstream>
using namespace std;
int main()
```

```
ofstream myfile;
myfile.open ("example.txt");
myfile << "Writing this to a file.\n";
myfile.close();
return 0;
}
```

Opening a file:-

A file must be opened before you can read from it or write to it. Either the ofstream or fstream object may be used to open a file for writing & ifstream object is used to open a file for reading purpose only. Following is the standard syntax for open() function which is a member of fstream, ifstream and ofstream objects.

void open (const char *filename, ios::openmode mode);

Here, the first argument specifies the name & location of the file to be opened & the second argument of the open() member function defines the mode in which the file should be opened.

Mode Flag.	Description:-
ios::app	Append mode. All output to that file to be appended to the end.
ios::ate	Open a file for output & move the read write control to the end of the file.
ios::in	Open a file for reading.
ios::out	Open a file for writing.
ios::trunc	If the file already exists, its contents will be truncated before opening the file.

You can combine 2 or more of these values by ORing

them together - For example if you want to open a file in write mode and want to truncate it in case it already exists, following will be the syntax:-

```
ofstream outfile;
outfile.open("file.dat", ios::out | ios::trunc);
```

Similar way, you can open a file for reading and writing purpose as follows:-

```
fstream afile;
afile.open("file.dat", ios::out | ios::in);
```

Closing a file:- When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory & close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination. Following is the standard syntax for close() function, which is a member of fstream, ifstream, and ofstream objects.

```
void close();
```

Write to a file:-

While doing C++ programming you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an ofstream or fstream object instead of the cout object.

Writing operations on text files are performed in the same way we operated with cout.

// writing on a text file:-

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
ofstream myfile ("example.txt");
if (myfile.is_open())
{
myfile << "This is a line.\n";
myfile << "This is another line.\n";
myfile.close();
}
else cout << "Unable to open file";
return 0;
}
```

Reading from a file:-

You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an ifstream or fstream object instead of the cin object.

Reading from a file can also be performed in the same way that we did with cin:-

```
// reading a text file.
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
  string line;
```

```
ifstream myfile ("example.txt");
if (myfile.is_open())
{
  while (getline (myfile,line))
  {
    cout << line << "\n";
  }
  myfile.close();
}
else cout << "Unable to open file";
return 0;
}
```

This last example reads a text file and prints out its content on the screen. We have created a while loop that reads the file line by line, using getline. The value returned by getline is a reference to the stream object itself, which when evaluated as a boolean expression (as in this while-loop) is true if the stream is ready for more operations, and false if either the end of the file has been reached or if some error occurred.

Text files:-

Text file streams are those where the ios::binary flag is not included in their opening mode. These files are designed to store text and thus all values that are input or output from/to them can suffer some formatting transformations, which do not necessarily correspond to their literal binary value.

Algorithm:
Step 1: Start
Step 2: Include required headers file like iostream (for keyboard input and output), fstream (for file reading & writing) and cstdlib (for exit()).
Step 3: In main(), read name of file from user.
Step 4: Open file using ofstream from writing.
Step 5: If file does not exist, new file will be created otherwise existing file will be overwritten.
Step 6: If any error occurs in creating or opening file, exit.
Step 7: Otherwise, open file.
Step 8: Read string from user.
Step 9: Check if given termination string is entered. (Here we will use "D" to end reading contents from keyboard)
Step 10: Otherwise, read line from keyboard and store in file using <<.
Step 11: Close the file.
Step 12: Open the same file for reading.
Step 13: Check if file opened for reading or display error message and exit.
Step 14: In while loop, start reading from file line by line and display o/p on terminal.
Step 15: If end of file is reached, exit loop.
Step 16: End program.
Step 17: Stop.

Conclusion:- Hence, we have studied & learnt various file handling operations and methods available in fstream header.

```
//program to create txt file at given destination and perform
operations
//Vrushil Soni-S1612037
#include<iostream>
#include<fstream>
#include<string.h>
using namespace std;

int main()
{
      char ch;
      string sample;
      ofstream o;
      o.open("/home/vrushil/Desktop/samplefile.txt");

      cout<<"Enter a line to store in the text file :  ";
      getline(cin,sample);
      o<<sample;
      o.close();


      ifstream i;
      i.open("/home/vrushil/Desktop/samplefile.txt");
      getline(i,sample);
      cout<<sample<<endl;
      cout<<"Do you want to close the file?"<<endl;
      cin>>ch;
      if(ch=='y'||ch=='Y')
            i.close();



      if(i.is_open())
            cout<<"File is open!please close the file to avoid
data loss!!"<<endl;
      else
            cout<<"File is closed!!";
      return 0;

}//end of program

//output

/*Enter a line to store in the text file :  This is sample
file created by me.
This is sample file created by me.
Do you want to close the file?
y
File is closed!!*/
```

S1612037

Experiment No 11.

Title: Demonstrate function template for sorting algorithm.

Objectives:- ① To learn and understand templates.
② To demonstrate function template for sorting using selection sort.

Problem Statement:-
Write a function template selection sort. Write a program that inputs, sorts and outputs an integer array and a float array.

Outcomes:- ① Students will be able to learn and understand working & use of function template.
② Students will be able to demonstrate function template for selection sort.

Hardware requirements:-
Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:-
64 bit Linux / Windows Operating System, G++ compiler.

Theory:- Templates are a feature of C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one. Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept. There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector <int> or vector <string>.

You can use template to define functions as well as classes, let us see how do they work.

Function Template:- The general form of a template function definition is shown here:-

```
template <typename type> ret-type func-name (parameter list)
{
   // body of function.
}
```

Here, type is a placeholder name for a data type an used by the function. This name can be used within the function definition.
A function template behaves like a function except that the template can have arguments of many different types (see example). In other words, a function template represents a family of functions. The format for declaring function templates with type parameters is:-

template < class identifier > function_declaration;
template < typename identifier > function_declaration;

Both expressions have the same meaning and behave in exactly the same way. The latter form was introduced to avoid confusion, since a type parameter need not be a class. (it can also be a basic type such as int or double).

For example, the C++ Standard Library contains the function template max(x, y) which returns the larger of x and y. The function template could be defined like this:-

```
template < typename T>
{
   return a>b ? a:b;
}
```

This single function definition works with many data types. The usage of a function template saves space in the source code file in addition to limiting changes to one function description making the code easier to read.

A template does not produce smaller object code though, compared to writing separate functions for all the different data types used in a specific program. For example, if a program uses both an int and a double version of the max() function template shown above, the compiler will create an object code version of max() that operates on int arguments and an object code version that operates on double arguments. The compiler output will be identical to what would have been produced if the source code had contained 2 separate non-templated versions of max(), one written to handle int and one written to handle double.

Selection sort is a simple sorting algorithm. This sorting algorithm is a in-place comparison based algorithm in which the list is divided into 2 parts, sorted part at left and unsorted part at right end. Initially sorted part is empty and unsorted part is entire list.

Smallest element is selected from the unsorted array and swapped with the leftmost element and that element becomes part of sorted array. This process continues moving unsorted array boundary by one element to the right.
This algorithm is not suitable for large data sets as its average & worst case complexity are of $O(n^2)$ where n are no. of items.

Step 1: Set Min to location 0.
Step 2: Search the minimum element in the list.
Step 3: Swap with value at location MIN.
Step 4: Increment MIN to point to next element.
Step 5: Repeat until list is sorted.

Algorithm:-

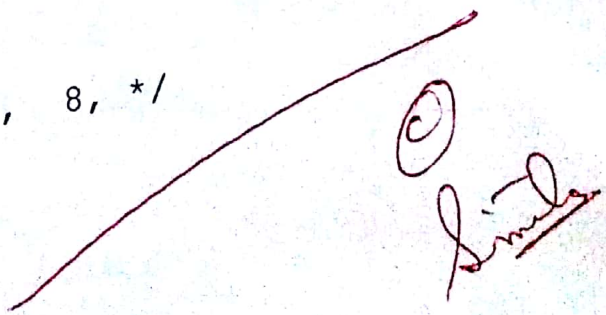
1. Define a function template with name input() and 2 arguments, one argument for array to sort and other for size of array.
2. Accept size number of elements from user for sorting in input().
3. Define the other function template display() with 2 arguments, one pointer of array to sort and other number of elements to sort.
4. Print sorted elements stored in array
5. Define another function template sort() to implement selection sort for generic datatype.
6. Declare a variable min to hold index position of smallest element in array to sort.
7. Define for loop with loop variable "i" from 0 to size, incremented by one.

(8). Assign first index position to min variable i.e
(9). Define second for loop with variable "j" st
from "i+1" to size, incremented by one.
(10). In inner for loop, check if element at index
position "j" is smaller than element at index
position "min".
(11). If condition is true, assign min as j else
(12). In outer for loop, swap element at index po
"i" with element at index position at "min".
(13). In main() method, declare variable size to
accept size of array.
(14). Input number of elements to sort and dec
an array of given size.
(15). Use int as datatype of array.
(16). Call input() function and pass array name and
size as argument.
(17). Call sort() function and pass array name
and size as argument to display sorted array
(18). Call display() function and pass array name
and size as argument to display sorted
array.
(19). Repeat steps from 15 to 19 with datatype
float.
(20). End program.

Conclusion: Hence, we demonstrated use of function
template for selection sort.

```
//program to show implementation of selection sort
//Vrushil Soni-S1612037
#include<iostream>
using namespace std;

template <class T>
void get_item (T a[],int n)
{
 for(int i=0;i<n;i++)
  {
    cout<<"\nEnter Elements:\n";
     cin>>a[i];
  }
 }

template <class T>
void s_sort(T a[],int n)
{
    T temp, min;
    int loc;

 for(int i=0;i<n;i++)
    {
        min=a[i];
        loc=i;
        for(int j=i+1;j<n;j++)
        {
            if(min>a[j])
            {
                min=a[j];
                loc=j;
            }
         }

         temp=a[i];
         a[i]=a[loc];
         a[loc]=temp;
     }
 }

 template <class T>
 void display (T a[],int n)
 {
  cout<<"\nAfter Sorting\n";
  for(int i=0;i<n;i++)
  {
   cout<<" "<<a[i]<<", ";
  }
 }


 int main()
 {
    int a[100],i,n;
     cout<<"Enter The number of Element:\n";
     cin>>n;
```

```
    get_item<int> (a,n);
    s_sort<int>(a,n);
    display<int>(a,n);

    float b[100],j,m;
    cout<<"Enter The number of Element:\n";
    cin>>m;
    get_item<float> (b,m);
    s_sort<float>(b,m);
    display<float>(b,m);

    char c[100];
    int o;
    cout<<"Enter The number of Element:\n";
    cin>>o;
    get_item (c,o);
    s_sort(c,o);
    display(c,o);


     return 0;
}//end of program

//output

/*Enter The number of Element:
5

Enter Elements:
2

Enter Elements:
1

Enter Elements:
8

Enter Elements:
0

Enter Elements:
3

After Sorting
 0,  1,  2,  3,  8, */
```



Experiment No: 12

Title:- Demonstration of Singly, Doubly and Circular Linked List using STL.

Objectives:- 1) To learn and understand concepts of standard Template Library.
2) To demonstrate STL for implementation of singly, doubly & circular linked list.

Problem statement: Write C++ program using STL for implementation of Singly, Doubly and Circular Linked list.

Outcomes:- 1) Students will be able to learn and understand concepts of STL.
2) Students will be able to demonstrate various operations on singly, doubly & circular linked list using STL.

Hardware requirements: Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard disk or more.

Software requirements: 64 bit Linux / Windows OS, C++ compiler.

Theory:- The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provide general purpose templatized classes & functions that implement many popular & commonly used

algorithms and data structures like vectors, lists, queues, and stacks. At the core of C++ STL are following 3 well-structured components:-

Containers: These are used to manage collection of objects of a certain kind. There are several different types of containers like deque, list, vector, etc. Algorithms :- Algorithms act on containers. They provide the means by which you will perform initialization, sorting, searching, and transformation of the contents of containers; ~~iterators~~.

Iterators:- are used to step through the elements of collections of objects. These collections may be containers or subsets of containers.

Standard Containers:-

A container is a holder object that stores a collection of other objects (its elements). They are implemented as class templates, which allows a great flexibility in the types supported as elements. The container manages the storage space for its elements & provides member functions to access them, either directly or through iterators (reference objects with similar properties to pointers).

Containers replicate structures very commonly used in programming: dynamic arrays, queues, stacks, heaps, linked lists, trees, associative arrays.

Many containers have several member functions in common, & share functionalities. The decision of which type of container to use for a specific need does not generally depend only on the functionality offered by the container, but also on the efficiency of some of its member (complexity). This is especially true for sequence containers, which offer different trade-offs in complexity between inserting/removing elements & accessing them.

Stack, queue & priority-queue are implemented as container adaptors. Container adaptors are not full container classes, but classes that provide a specific interface relying on an object of one of the container classes to handle the elements. The underlying container is encapsulated in such a way that its elements are accessed by the members of the container adaptor independently of the underlying container class used.

List :-

List containers are implemented as doubly-linked lists; Doubly linked list can store each of the elements they contain in different and unrelated storage locations. The ordering is kept internally by the association to each element a link to the element preceding it & a link to element following it.

They are very similar to forward_list. The main difference being that forward-list objects are single-linked lists, and thus they can only be iterated forwards, in exchange for being somewhat smaller & more efficient.

Compared to other base standard sequence containers (array, vector & deque), lists perform generally better in inserting, extracting & moving elements in any position within the container for which an iterator has already been obtained, and therefore also in algorithms that make intensive use of these, like sorting algorithms.

The main drawback of lists & forward_lists compared to these other sequence containers is that they lack direct access to the elements by their position; for example, to access the 6th element in a list, one has to iterate from a known position (like the beginning or the end) to that position, which takes linear time in the distance between these. They also consume some extra memory to keep the linking information associated to each element (which may be an important factor for large lists of small-sized elements).

Container properties:-

Sequence

Elements in sequence containers are ordered in a strict linear sequence. Individual elements are accessed by their position in the sequence.

Doubly-linked list.

Each element keeps information on how to locate the next & the previous elements, allowing constant time insert & erase operations before or after a specific element (even of entire ranges), but no direct random access.

Allocator -aware:-
The container uses an allocator object to dynamically handle its storage needs.
Functions used with list:-
front() - Returns reference to 1st element in list.
back() - Returns reference to last element in list.
push_front(g) - Adds a new element 'g' at begin of list.
push_back(g) - Adds a new element 'g' at the end of list.
pop_front() - removes the 1st element of list, & reduces size of list by 1.
pop_back() - Removes the last element of list, & reduces size of the list by 1.
begin() - returns an iterator pointing to the 1st element of list.
end() - returns an iterator pointing to the theoretical last element which follows the last element.
insert() - Inserts new elements in list before element at a specified position.
erase() - removes all the elements from the list which are equal to given element.
reverse() - reverse the list.
size() - returns the no of elements in list.
sort() - Sorts the list in increasing order.

Algorithm:-
Step 1: Include header files iostream, list.
Step 2: Use namespace std for templates.
Step 3: Start with main method.
Step 4: Declare an object of singly, doubly & circular linked list class with int as datatype.
Step 5: Declare a variable, choice, to accept user choice for main menu.

Step 6: Declare a variable, element, to store an el
insert in list.
Step 7: Declare a variable, ch, to store user choi
continue with other main menu.
Step 8: Print menu with options as insert elemen
delete element, check size, display contents o
exit.
Step 9: Accept choice of user from menu option.
Step 10: In switch case, define cases with each ch
Step 11: For choice 1, accept elemet from user. Use pu
front() or push_back() algorithm to ins
Step 12: For choice 2, call pop_back or pop_front()
algorithm to delete element.
Step 13: In choice 3, use size() algorithm of singly, do
& circular linked list class to display si
Step 14: For choice 4, use while loop with empty() algo
to display elements of lists.
Step 15: In choice 5, use exit functions to end prog

Conclusion:- Hence, we have studied & demonstrated use of STL class to implement singly, doubly & circular linked list and various operation.

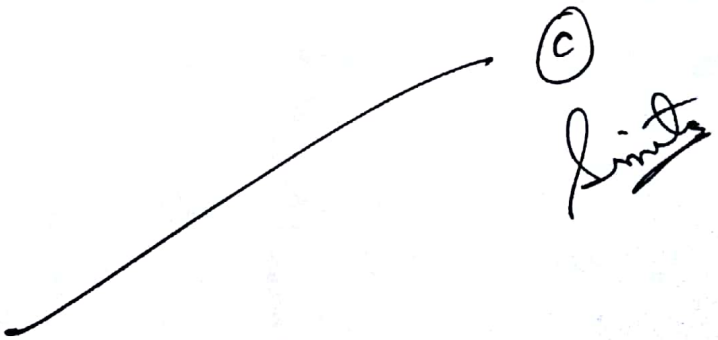
(C)

```
/*NAME:- Vrushil Soni
CLASS:- SE(c)
ROLL No:-S1612037
SUB:- OOP

                              Assignment No:- C_20(List)
Write C++ program using STL for implementation of Singly,
doubly and circular linked list.
*/

#include<iostream>
#include<list>
using namespace std;
class stlist
{
    list<int>mylist,mylist2;
    list<int>::iterator it;
public:
    void checkempty();
    void insertele();
    void makelist();
    void mergerlist();
    void reverselist();
    void sortlist();
    void uniquelist();
    void display();
};
void stlist::checkempty()
{
    if(mylist.empty())
        cout<<"List is empty\n";
    else
        cout<<"List not empty"<<endl;
}
void stlist::makelist()
{
    int count;
    int a;
    cout<<"Enter the number of elements in your list"<<endl;
    cin>>count;
    cout<<"Enter the elements:"<<endl;
    for(int i=0;i<count;i++)
    {
        cin>>a;
        mylist.push_back(a);
    }
}
void stlist::insertele()
{
    int pos,num;
    cout<<"Enter the position at which you want to enter
element:"<<endl;
    cin>>pos;
    cout<<"Enter the element you want to enter:"<<endl;
    cin>>num;
    it=mylist.begin();
    for(int i=0;i<pos;i++)
```

```
        ++it;
    mylist.insert(it,num);
}
void stlist::display()
{
    cout<<"List:"<<endl;
    for(it=mylist.begin();it!=mylist.end();it++)
        cout<<*it<<endl;
}
void stlist::sortlist()
{
    cout<<"Sorted list:"<<endl;
    mylist.sort();
}
void stlist::reverselist()
{
    cout<<"Reversed list:"<<endl;
    mylist.reverse();
}
void stlist::uniquelist()
{
    mylist.unique();
}
void stlist::mergerlist()
{
    int count;
        int a;
        cout<<"Enter the number of elements in your second
list"<<endl;
        cin>>count;
        cout<<"Enter the elements:"<<endl;
        for(int i=0;i<count;i++)
        {
            cin>>a;
            mylist2.push_back(a);
        }
        mylist.merge(mylist2);
}
int main()
{
    stlist obj;
    int choice;
    char c='y';
    do
    {
    cout<<"1. Enter elements of list\n2. Check if list is
empty\n3. Insert element in list\n4. Sort list\n5. Print list
in reverse order\n6. Delete repeated elements in list\n7.
Merge 2 lists\nEnter choice:"<<endl;
    cin>>choice;
    switch(choice)
    {
    case 1:
        obj.makelist();
        break;
    case 2:
        obj.checkempty();
```

```
            break;
        case 3:
            obj.insertele();
            obj.display();
            break;
        case 4:
            obj.sortlist();
            obj.display();
            break;
        case 5:
            obj.reverselist();
            obj.display();
            break;
        case 6:
            obj.uniquelist();
            obj.display();
            break;
        case 7:
            obj.mergerlist();
            obj.display();
            break;
        default:
            cout<<"Wrong choice"<<endl;
        }
        cout<<"Do you wish to continue? Y or N\n";
        cin>>c;
        }while(c=='y'||c=='Y');
        return 0;
}/*

OUTPUT:-
1. Enter elements of list
2. Check if list is empty
3. Insert element in list
4. Sort list
5. Print list in reverse order
6. Delete repeated elements in list
7. Merge 2 lists
Enter choice:
1
Enter the number of elements in your list
2
Enter the elements:
2
3
Do you wish to continue? Y or N
Y
1. Enter elements of list
2. Check if list is empty
3. Insert element in list
4. Sort list
5. Print list in reverse order
6. Delete repeated elements in list
7. Merge 2 lists
Enter choice:
2
List not empty
```

```
Do you wish to continue? Y or N
Y
1. Enter elements of list
2. Check if list is empty
3. Insert element in list
4. Sort list
5. Print list in reverse order
6. Delete repeated elements in list
7. Merge 2 lists
Enter choice:
3
Enter the position at which you want to enter element:
1
Enter the element you want to enter:
4
List:
2
4
3
Do you wish to continue? Y or N
Y
1. Enter elements of list
2. Check if list is empty
3. Insert element in list
4. Sort list
5. Print list in reverse order
6. Delete repeated elements in list
7. Merge 2 lists
Enter choice:
*/
```



Experiment No: 13

Title: Demonstration of STL.

Objectives:-
1) To learn & understand concepts of Standard Template Library.
2) To demonstrate STL for implementation of stack & queue operations.

Problem Statement: Write C++ program using STL for implementation of Stack & queue.

Outcomes:-
1) Students will be able to learn & understand concepts of STL.
2) Students will be able to demonstrate various operations on stack & Queue using STL.

Hardware requirements: Any CPU with pentium processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software requirements: 64 bit Linux / windows Operating System, C++ compiler.

Theory:- The C++ STL is a powerful set of C++ template classes to provide general-purpose templatized classes and functions that implement many popular and commonly used algorithms and data structures like vectors, lists, queues & stacks. At the core of C++ STL are following

3 well-structured components.

Containers: Containers are used to manage collec of objects of a certain kind. There are several di types of containers like deque, list, vector, map,

Algorithms: Algorithms acts on containers. They provide the means by which you will perform initialization, sorting, searching, and transform of the contents of containers

Iterators: Iterators are used to step through the ele of collections of objects. These collections may be containers or subsets of containers.

Standard containers:-

A container is a holder object that stores a collecti of other objects (its elements). They are implement as class templates, which allows a great flexibility in the types supported as elements.

The container manages the storage space for its elements & provides member functions to access the either directly or through iterators (reference objects with similar properties to pointers).

Containers replicate structures very commonly used in programming dynamic arrays, queues, stacks, heaps, linked lists, trees, associative arrays.

Many containers have several member functions in common, and share functionalities. The decision

of which type of container to use for a specific need does not generally depend only on the functionality offered by the container, but also on the efficiency of some of its members (complexity). This is especially true for sequence containers, which offer different trade-offs in complexity between inserting / removing elements & accessing them.

Stack, queue & priority-queue are implemented as container adaptors. Container adaptors are not full container classes, but classes that provide a specific interface relying on an object of one of the container classes to handle the elements. The underlying container is encapsulated in such a way that its elements are accessed by the members of the container adaptor independently of the underlying container class used.

Container class templates.
Sequence containers.

array : Array class.
vector: vector.
deque: Double ended queue.
list : List.

Container adaptors:-
stack :- LIFO stack.
queue: FIFO queue.
priority_queue : Priority queue.

Associative containers:-
set: Set.
multiset: multiple-key set.
map: Map.
multimap: multiple-key map.
Containers are implemented via template class definitions. This allows us to define a group of elements of the required type. For example, if we need an array in which the elements are simply integers (as in the example above), we would declare it as:

vector <int> values;

(Yes, the name of the container is to represent arrays, i.e. vector - it follows the mathematical idea that a vector group of values in a multi-dimensional space)

The iterator is container-specific. That is, the iterator operations depend on what type of container we are using. For that reason, the iterator class definition is inside the container class; this has a good side effect, which is that the syntax to declare an iterator suggests that the definition type belongs in the ~~context~~ context of the container definition. Thus, we would declare an iterator for the values object as follows:-

vector <int>::iterator current;

With this, we are almost ready to translate the example shown above to find the highest value in an array, using STL tools. I will first present the

implementation, & then explain the details.

```
vector<int>::iterator current = values.begin();
int high = *current++;
while (current != values.end())
{
    if (*current > high)
    {
        high = *current;
    }
    current++;
}
```

Here, instead of declaring a pointer to int and initializing it pointing to the 1st element of array, we declare an iterator for a vector of integers (i.e., an element that will "point" to integers contained in a vector object), and we initialize it "pointing" to the first element in values. Given that the values is now an object & not a standard array, we need to ask that object to give us a "pointer" to the first element (more exactly, to give us an iterator that is "pointing" to the 1st element).

This is done with the member-function begin(), which is provided by all the containers. The value of the iterator pointing to one past the end of the array is also provided by the container object, via the member fn end().

Other than these 2 details, the rest is identical.

Following is the algorithm to find the highest value in a linked list of integers is coded as follows using STL tools:-

```
list <int> :: iterator current = values.begin();
int high = *current++;
while (current != values.end())
{
  if (*current > high)
    {
    high = *current;
    }
  current++;
}
```

In particular, most containers provide member-fu to insert or remove elements from the ends. The operations refer to operations performed at beginning (first element), and the "back" operations at the end (last element). The following member functions are provided for most containers.

1) push_front: inserts elements before the first (not available for vector).
2) pop_front: removes the 1st element (not available for vector).
3) push_back: removes the last element.

Also, most containers provide the following member functions:-

1) empty: Boolean indicating if the container is empty.
2) size: returns the no of elements.
3) insert: inserts an element at a particular position
4) erase: removes an element at a particular position
5) clear: removes all the elements.
6) resize: resizes the container.
7) front: Returns a reference to first element.
8) back: returns a reference to last element.

The vector & deque containers provide subscripting access to elements, and subscripting access with bounds-checking:

1] [] Subscripting access without bounds checking.
2] at subscripting access with bounds checking.

LIFO stack:- Stacks are a type of container adaptor, specifically designed to operate in a LIFO context (last-in-first-out), where elements are inserted and extracted only from one end of the container.

Stacks are implemented as container adaptors, which are classes that use an encapsulated object of a specific container class as its underlying container, providing a specific set of member functions to access its elements. Elements are pushed/popped from the 'back' of the specific container, which is known as top of the stack.

FIFO Queue:- Queues are a type of container adaptor, which are classes that use an encapsulated object of a specific container class as its underlying container, providing a specific set of member functions to access its elements. Elements are pushed into the 'back' of the specific container & popped from its 'front'.

The std container classes deque and list fulfill these requirements. By default, if no container class is specified for a particular queue class instantiation, the std container deque is used.

Algorithm Step 1: Start.
Step 2: Include header files iostream, stack & queue
Step 3: Use namespace std for templates. Start main.
Step 4: Declare an object of stack class with int as
datatype.
Step 5: Declare an object of queue class with int as d
Step 6: Declare a variable, ch, to store user choice to conti
other menu option.
Step 7: Declare a variable, element to store an element
insert in stack & queue.
Step 8: Declare a variable, choice to accept user c
for main menu.
Step 9: Print menu with options as insert element in stac
& queue, delete element, check size, display
1st element, display contents & exit.
Step 10: Accept choice from user from menu option.
Step 11: In switch case, define cases with each choice.
Step 12: For choice 1, accept an element from user. Use push
algorithm to insert element at top of stack & at
rear end of queue.
Step 13: For choice 2, call pop() algorithm to delete top ele
from stack & front element from queue.
Step 14: In choice 3, use size() algorithm of stack &
queue class to display size.
Step 15: In choice 4, display top element of stack, front
back element of queue.
Step 16: For choice 5, use while loop, with empty()
algorithm to display top element of stack &
pop() top element. Also, use second while
loop with empty() to display front element of
queue and pop() element of queue.

Step 17: In choice 6, use exit function to end the program.

Step 18: Stop.

lusion:- Hence, we have studied and demonstrated use of ST class stack and queue to implement various operations.

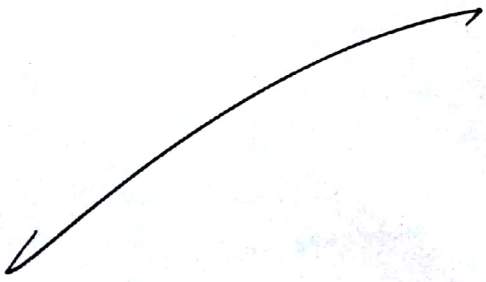
```
//program to create stack and queue using SLL.
//Vrushil Soni-S1612037
#include<iostream>
#include<string.h>
#include<list>
using namespace std;
#define MAX 100
//class to create queue using sll
class Queue
{
      public :
            int a[MAX],size_q;
            list <int> queue;
            list <int> :: iterator itr;

            void push()
            {
                  cout<<"ENTER THE SIZE FOR YOUR QUEUE :
"<<endl;
                  cin>>size_q;
                  cout<<"ENTER THE ELEMENTS IN THE QUEUE :
"<<endl;
                  for(int i=0;i<size_q;i++)
                  {
                        cin>>a[i];
                        queue.push_back(a[i]);
                  }
            }

            void displayqueue()
            {
                  cout<<"\n THE ELEMENTS IN THE QUEUE ARE  :
"<<"\n";
                  for(itr=queue.begin() ; itr!=queue.end() ;
itr++)
                  {
                        cout<<"|"<<*itr<<"|";
                  }
            }

            void pop()
            {
                  itr=queue.begin();
                  queue.pop_front();
                  cout<<"\n THE ELEMENT POPPED OUT IS : "<<*itr
<<endl;
            }

};
//class to create stack using sll.
class Stack
{
      public :
            int a[MAX],size_s;
            list <int> stack;
            list <int> :: iterator itr;
```

```
        void push()
        {

<<endl;         cout<<"ENTER THE CAPACITY FOR YOUR STACK :
                cin>>size_s;
<<endl;         cout<<"ENTER THE ELEMENTS IN THE STACK :
                for(int i=0;i<size_s;i++)
                {
                    cin>>a[i];
                    stack.push_back(a[i]);
                }
        }

        void displaystack()
        {
                cout<<"\n THE CONTENTS OF STACK ARE : "<<"\n";
itr++)          for(itr=stack.begin() ; itr!=stack.end() ;
                {
                    cout<<"____"<<endl;
                    cout<<*itr<<endl<<"____"<<endl;
                }
        }

        void pop()
        {
                itr=stack.end();
                itr--;
                stack.pop_front();
                cout<<"\n THE ELEMENT POPPED FROM YHE STACK IS
: "<<*itr<<endl;
            }

};

int main()
{
    Queue q;
    Stack s;
    int ch,size_q;
    char ans;
    do
    {
        cout
<<"*********************************MENU*************************
**********"<<endl;
        cout<<"\n 1. ADD ELEMENT IN QUEUE \n 2. DELETE
ELEMENT IN QUEUE \n 3. DISPLAY QUEUE \n 4.ADD ELEMENT IN STACK
\n 5.DELETE ELEMENT IN STACK \n"
        <<"6.DISPLAY STACK \n"<<endl;
        cout<<"\n ENTER YOUR CHOICE  : "<<endl;
        cin>>ch;
        switch(ch)
```

```
{
        case 1 :

                q.push();
                break;
        break;
        case 2 :

                q.pop();
                break;
        break;
        case 3 :

                q.displayqueue();
                break;
        case 4:
                s.push();
                break;
        case 5:
                s.pop();
                break;
        case 6:
                s.displaystack();
                break;
        default:
                cout<<"!!!!!!!!!!!!!!!!
INVALID!!!!!!!!!!!!!!!!!!!!!!!!"<<endl;
        }
        cout<<"\n DO YOU WANT  TO CONTINUE?Y/N?"<<endl;
        cin>>ans;
    }while(ans=='Y' || ans=='y');
    return 0;
}//end of program

//output

/
**********MENU**********
********

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
1
ENTER THE SIZE FOR YOUR QUEUE :
5
ENTER THE ELEMENTS IN THE QUEUE :
1
2
3
```

```
4
5
 DO YOU WANT  TO CONTINUE?Y/N?
y
**********************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
3

 THE ELEMENTS IN THE QUEUE ARE  :
|1||2||3||4||5|
 DO YOU WANT  TO CONTINUE?Y/N?
y
*******************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
2

 THE ELEMENT POPPED OUT IS : 1

 DO YOU WANT  TO CONTINUE?Y/N?
y
*******************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
3

 THE ELEMENTS IN THE QUEUE ARE  :
|2||3||4||5|
```

```
 DO YOU WANT  TO CONTINUE?Y/N?
y
**********************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
4
ENTER THE CAPACITY FOR YOUR STACK :
5
ENTER THE ELEMENTS IN THE STACK :
1
2
3
4
5

 DO YOU WANT  TO CONTINUE?Y/N?
y
**********************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
6

 THE CONTENTS OF STACK ARE :

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



```
 DO YOU WANT  TO CONTINUE?Y/N?
y
**********************************MENU**************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
5

 THE ELEMENT POPPED FROM YHE STACK IS : 5

 DO YOU WANT  TO CONTINUE?Y/N?
y
**********************************MENU**************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
6

 THE CONTENTS OF STACK ARE :

_____
2
_____
_____
3
_____
_____
4
_____
_____
5
_____

 DO YOU WANT  TO CONTINUE?Y/N?
n
**********************************MENU**************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
```

```
5.DELETE ELEMENT IN STACK
6.DISPLAY STACK

 ENTER YOUR CHOICE  :
1
ENTER THE SIZE FOR YOUR QUEUE :
5
ENTER THE ELEMENTS IN THE QUEUE :
1
2
3
4
5

 DO YOU WANT  TO CONTINUE?Y/N?
y
*******************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
3

 THE ELEMENTS IN THE QUEUE ARE  :
|1||2||3||4||5|
 DO YOU WANT  TO CONTINUE?Y/N?
y
*******************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
2

 THE ELEMENT POPPED OUT IS : 1

 DO YOU WANT  TO CONTINUE?Y/N?
y
*******************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
```

```
2. DELETE ELEMENT IN QUEUE
3. DISPLAY QUEUE
4.ADD ELEMENT IN STACK
5.DELETE ELEMENT IN STACK
6.DISPLAY STACK

 ENTER YOUR CHOICE  :
3

 THE ELEMENTS IN THE QUEUE ARE  :
|2||3||4||5|
 DO YOU WANT  TO CONTINUE?Y/N?
y
**********************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
4
ENTER THE CAPACITY FOR YOUR STACK :
5
ENTER THE ELEMENTS IN THE STACK :
1
2
3
4
5

 DO YOU WANT  TO CONTINUE?Y/N?
y
**********************************MENU*****************************
*******

 1. ADD ELEMENT IN QUEUE
 2. DELETE ELEMENT IN QUEUE
 3. DISPLAY QUEUE
 4.ADD ELEMENT IN STACK
 5.DELETE ELEMENT IN STACK
6.DISPLAY STACK


 ENTER YOUR CHOICE  :
6

 THE CONTENTS OF STACK ARE :
____
1
____
____
```

2
3
4
5

DO YOU WANT TO CONTINUE?Y/N?
y
**********MENU**********
1. ADD ELEMENT IN QUEUE
2. DELETE ELEMENT IN QUEUE
3. DISPLAY QUEUE
4.ADD ELEMENT IN STACK
5.DELETE ELEMENT IN STACK
6.DISPLAY STACK

ENTER YOUR CHOICE :
5

THE ELEMENT POPPED FROM YHE STACK IS : 5

DO YOU WANT TO CONTINUE?Y/N?
y
**********MENU**********
1. ADD ELEMENT IN QUEUE
2. DELETE ELEMENT IN QUEUE
3. DISPLAY QUEUE
4.ADD ELEMENT IN STACK
5.DELETE ELEMENT IN STACK
6.DISPLAY STACK

ENTER YOUR CHOICE :
6

THE CONTENTS OF STACK ARE :
2
3
4

```
 DO YOU WANT  TO CONTINUE?Y/N?
n*/
```

Experiment No: 14

Title:- Demonstration of STL.

Objectives:-
1) To learn & understand concepts of standard Template Library.
2) To demonstrate STL for implementation of deque operations.

Problem statement: Write CPP program using STL for Dqueue (Double Ended Queue).

Outcomes:
1) Students will be able to learn and understand concepts of STL.
2) Students will be able to demonstrate various operations on deque using STL.

Hardware requirements: Any CPU with Pentium Processor or similar, 256 MB ram or more, 1 GB hard disk or more.

Software requirements: 64 bit Linux / windows Operating system, C++ compiler.

Theory:- The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provide general-purpose templatized classes & functions that implement many popular and commonly used algorithms and data structures like vectors, lists, queues, stacks, etc.

At the core of the C++ STL are following 3 well-structured components:-

Containers:- Containers are used to manage collection of objects of a certain kind. There are several different types of ~~conte~~ containers like deque, list, vector, map, etc.

Algorithms:- Algorithms act on containers. They provide means by which you will perform initialization, sorting, searching, and transforming of contents of containers.

Iterators:- Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers.

Containers: Standard Containers.

A container is a holder object that stores a collection of other objects (its elements). They are implemented as class templates, which allows a great flexibility in the types supported as elements. The container manages the storage space for its elements and provides member functions to access them, either directly or through iterators (reference objects with similar properties to pointers).

Containers replicate structures very commonly used in programming dynamic arrays, queues, stacks, heaps, linked list, trees, associative arrays.

Many containers have several member functions in common, and share functionalities. The decision of which type of container to use for a specific need does not generally depend only on the functionality offered by the containers, but also on the efficiency of some of its members (complexity). This is especially true for sequence containers, which offer different trade-offs in complexity between inserting/removing elements and accessing them.

Stack, queue, and priority-queue are implemented as container adaptors. Container adaptors are not full container classes, but classes that ~~provot~~ provide a specific interface relying on an object of one of the container classes to handle the elements. The underlying container is encapsulated in such a way that its elements are accessed by the members of the container adaptor independently of the underlying container class used.

Containers are implemented via template class definitions. This allows ~~yo~~ us to define a group of elements of ~~the required type~~. For example, if we need an array in which the elements are simply integers (as in the example above), we would declare it as:

```
vector <int> values;
```

The iterator is container specific. That is, the iterator's operations depend on what type of

containers we are using. For that reason, the iterator class definition is inside the container class; this has a good side effect, which is that the syntax to declare an iterator suggests that the definition belongs in the context of the container definition. Thus we would declare an iterator for the values object as follows:

```
vector<int>::iterator current;
```

With this, we are almost ready to translate the example shown above to find the highest value in an array using the STL tools. I will first present the implementation, and then explain the details:

```
vector<int>::iterator current = values.begin();
int high = *current++;
while (current != values.end())
{
    if (*current > high)
    {
        high = *current;
    }
    current++;
}
```

Here, instead of declaring a pointer to int & initializing it pointing to 1st element of array, we declare an iterator for a vector of integers (i.e. an element that will "point" to integers contained in a vector object) & we initialize it 'pointing' to the first element in values. Given that values is now an object and not

a standard array, we need to ask that object to give us a "pointer" to the first element (more exactly, to give us an iterator that is "pointing" to the first element).

This is done with the member fn begin(), which is provided by all the container classes. The value of the iterator pointing to one past the end of the array is also provided by the container object, via the member-function end().

Other than these 2 details, the rest is identical.

Following is the algorithm to find the highest value in a linked list of integers is coded as follows using the STL tools:-

```
list <int> :: iterator current = values.begin();
int high = * current ++;
while (current != values.end())
{
  if (* current > high)
  {
    high = * current;
  }
  current ++;
}
```

Double ended queue :-

deque (usually pronounced as 'deck') is an irregular acronym of double-ended queue. Deque are sequence containers with dynamic sizes that

can be expanded or contracted on both ends.
Specific libraries may implement deques in different ways, generally as some form of dynamic arrays. But in any case, they allow for the individual elements to be accessed directly through random access iterators, with storage handled automatically by expanding & contracting the containers as needed. Therefore, they provide a functionality similar to vectors, but with efficient insertion & deletion of elements also at the beginning of the sequence, and not only at its end. But, unlike vectors, deques are not guaranteed to store all its elements in contiguous storage locations: accessing elements in a deque by offsetting a pointer to another element causes undefined behaviour.

Both vectors & deques provide a very similar interface & can be used for similar purposes, but internally both work in quite different ways: while vectors use a single array that needs to be occasionally reallocated for growth, the elements of a deque can be scattered in different chunks of storage, with the container keeping the necessary information internally to provide direct access to any of its elements in constant time & with a uniform sequential interface (through iterators). Therefore, deques are a little more complex internally than vectors, but this allows them to grow more efficiently under certain circumstances, especially with very long sequences, where reallocations become more expensive. For operations that involve frequent insertion or

removals of elements at positions other than the beginning or end, deques perform worse and have less consistent iterators.

Conclusion:-

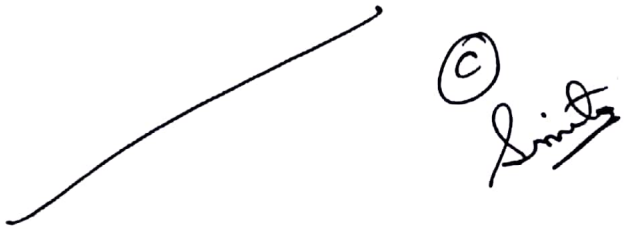
Hence, we studied & demonstrated use of deque to implement various operations.

```
//program to create a double ended linked list.
//Vrushil Soni-S1612037

#include<iostream>
#include<string.h>
#include<deque>
#define MAX 100
using namespace std;

class Dequeue
{
    public :
        int a[MAX],size_q;
        deque <int> ll;
        deque <int> :: iterator itr;

        void push_front()
        {
            cout<<"\n ENTER THE SIZE FOR YOUR DEQUEUE  :
"<<endl;
            cin>>size_q;
            for(int i=0;i<size_q;i++)
            {
                cin>>a[i];
                ll.push_front(a[i]);
            }
        }

        void push_back()
        {
            for(int i=0;i<size_q;i++)
            {
                cin>>a[i];
                ll.push_back(a[i]);
            }
        }

        void pop_back()
        {
            itr=ll.end();
            itr--;
            ll.pop_back();
            cout<<"\n THE ELEMENT POPPED OUT WAS : "<<*itr
<<endl;
        }

        void pop_front()
        {
            itr=ll.begin();
            ll.pop_front();
            cout<<"\n THE ELEMENT POPPED OUT WAS : "<<*itr
<<endl;
        }

        void displayqueue()
        {
```

```
"<<"\n";
itr++)
        {
        }
    }
};
int main()
{
    Dequeue obj;
    int ch;
    char ch1;
    do
    {
        cout<<"\n THE DEQUE IS AS FOLLOWS :
        for(itr=ll.begin() ; itr!=ll.end() ;
        cout<<"|"<<*itr<<"|";
        cout<<"\n 1. ADD ELEMENTS TO THE FRONT \n 2. ADD
ELEMENTS FROM BEHIND \n 3. DELETE AN ELEMENT FROM FRONT \n
4.DELETE AN ELEMENT FROM BEHIND \n 5. DISPLAY"<<endl;
        cout<<"\n ENTER YOUR CHOICE : ";
        cin>>ch;
        switch(ch)
        {
            case 1 :
                obj.push_front();
                break;
            case 2 :
                obj.push_back();
                break;
            case 3 : obj.pop_front();
                break;
            case 4 : obj.pop_back();
                break;
            case 5 : obj.displayqueue();
                break;
        }
        cout<<"\nDO YO WANT TO CONTINUE?Y/N?";
        cin>>ch1;
    }while(ch1=='Y' || ch1=='y');
    return 0;
}//end of program

//output

/* 1. ADD ELEMENTS TO THE FRONT
 2. ADD ELEMENTS FROM BEHIND
 3. DELETE AN ELEMENT FROM FRONT
 4.DELETE AN ELEMENT FROM BEHIND
 5. DISPLAY

 ENTER YOUR CHOICE : 1

 ENTER THE SIZE FOR YOUR DEQUEUE :
5
5
```

```
4
3
2
1
DO YO WANT TO CONTINUE?Y/N?y

 1. ADD ELEMENTS TO THE FRONT
 2. ADD ELEMENTS FROM BEHIND
 3. DELETE AN ELEMENT FROM FRONT
 4.DELETE AN ELEMENT FROM BEHIND
 5. DISPLAY

 ENTER YOUR CHOICE : 2
6
7
8
9
10

DO YO WANT TO CONTINUE?Y/N?y

 1. ADD ELEMENTS TO THE FRONT
 2. ADD ELEMENTS FROM BEHIND
 3. DELETE AN ELEMENT FROM FRONT
 4.DELETE AN ELEMENT FROM BEHIND
 5. DISPLAY

 ENTER YOUR CHOICE : 5

 THE DEQUE IS AS FOLLOWS :
|1||2||3||4||5||6||7||8||9||10|
DO YO WANT TO CONTINUE?Y/N?y

 1. ADD ELEMENTS TO THE FRONT
 2. ADD ELEMENTS FROM BEHIND
 3. DELETE AN ELEMENT FROM FRONT
 4.DELETE AN ELEMENT FROM BEHIND
 5. DISPLAY

 ENTER YOUR CHOICE : 3

 THE ELEMENT POPPED OUT WAS : 1

DO YO WANT TO CONTINUE?Y/N?y

 1. ADD ELEMENTS TO THE FRONT
 2. ADD ELEMENTS FROM BEHIND
 3. DELETE AN ELEMENT FROM FRONT
 4.DELETE AN ELEMENT FROM BEHIND
 5. DISPLAY

 ENTER YOUR CHOICE : 5

 THE DEQUE IS AS FOLLOWS :
|2||3||4||5||6||7||8||9||10|
DO YO WANT TO CONTINUE?Y/N?y
```

```
1. ADD ELEMENTS TO THE FRONT
2. ADD ELEMENTS FROM BEHIND
3. DELETE AN ELEMENT FROM FRONT
4.DELETE AN ELEMENT FROM BEHIND
5. DISPLAY

ENTER YOUR CHOICE : 4

THE ELEMENT POPPED OUT WAS : 10

DO YO WANT TO CONTINUE?Y/N?y

1. ADD ELEMENTS TO THE FRONT
2. ADD ELEMENTS FROM BEHIND
3. DELETE AN ELEMENT FROM FRONT
4.DELETE AN ELEMENT FROM BEHIND
5. DISPLAY

ENTER YOUR CHOICE : 5

 THE DEQUE IS AS FOLLOWS  :
|2||3||4||5||6||7||8||9|
DO YO WANT TO CONTINUE?Y/N?n*/
```



ASSIGNMENT NO: Mini Project

PROBLEM STATEMENT:

Tic-Tac- Toe gaming application using C++ Programming.

AIM:

To learn Tic-Tac-Toe algorithm.

FACILITIES (TOOLS USED): Linux Operating Systems, Turbo C++.

ALGORITHM:

1. Start
2. Decide who is first player or the second player.
3. Learn the specific moves associated with the board
4. Make a move that blocks opponent's attempt to win i. e. Counter-Attack.
5. Check winner
6. Display winner player name
7. Stop

ASSUMPTION:

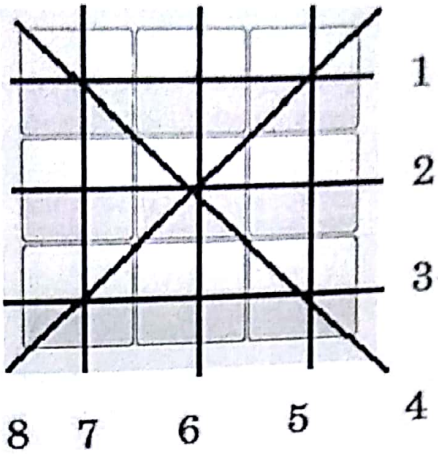
The given board is of 3x3.

THEORY:

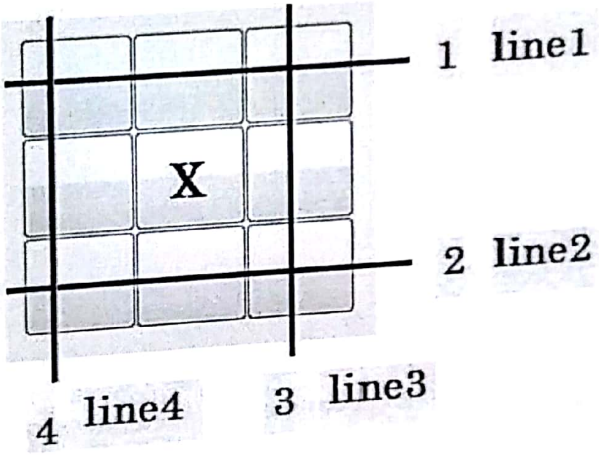
Tic-tac-toe is a well know game. Basically there is a 3x3 board and two players: one is X and the other is O. Learn the definitions and specific moves associated with the following words:

- Counter-Attack: Making a move that blocks your opponent's attempt to win.
- Center: The only square on a Tic Tac Toe board completely surrounded by other squares.
- Edge: A square bordering the center.
- Corner: A square bordered by 2 edge squares.
- Checkwin: Checks if either one of the players connected three blocks in row, column, or diagonal (8 possibilities).

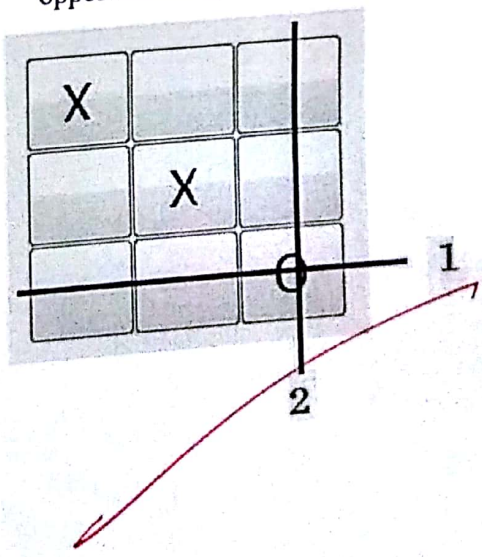
As the following figure shows, there are eight probabilities for connecting three blocks with the same symbol.



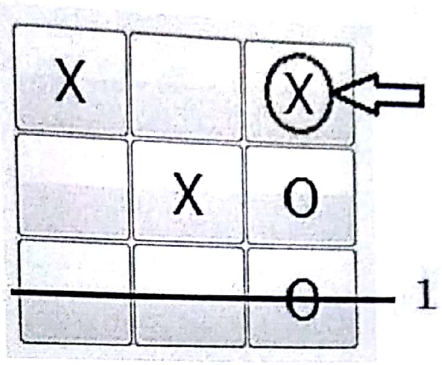
First step: Fill the center block. By this way I forbid the opponent from four opportunities and he still has another 4 as the following figure shows.



Second step: However the opponent plays, there is no danger till now and I have to decreased his number of opportunities. The best location for playing now is the corners as it decreases his opportunities by two in one turn.



Third step: Check the two lines (1 and 2) as the previous figure shows, and the one which contains two similar symbols; the computer should set the third with the other opposite symbol so I don't let the opponent win as the following figure shows.



Fourth step: Check the last available line for the opponent to win and check if he has two blocks set with his symbol and then check the third.

INPUT: A number represents a square of a 3x3 board.

OUTPUT: Game result

FAQ:

1) What is Tic-Tac-Toe algorithm?

2) Explain probabilities to win game using tic tac toe algorithm

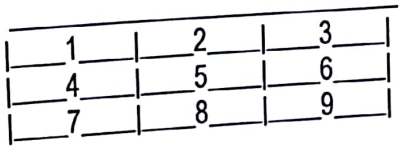
CONCLUSION:

Thus, we have designed tic-tac-toe game algorithm.

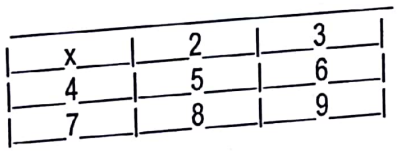
```
/*
NAME:- Vrushil Soni
CLASS:- S.E(c)
ROLL No:-S1612037
SUB:- OOP

Assignment :- MINI PROJECT(TIC-TOC-TOE)

Design and develop the Tic-Tac-Toe Game using C++

*/
#include<iostream>
using namespace std;
char square[10]={'0','1','2','3','4','5','6','7','8','9'};
char a[10],b[10];
void board()
{
        cout<<"\n\t"<<"     "<<"         "<<"         "<<"     ";

cout<<"\n\t"<<"|     "<<square[1]<<"     |     "<<square[2]<<"     |     "<<square[3]<<"     |"<
<endl;

cout<<"\t"<<"|     "<<square[4]<<"     |     "<<square[5]<<"     |     "<<square[6]<<"     |"<<e
ndl;

cout<<"\t"<<"|     "<<square[7]<<"     |     "<<square[8]<<"     |     "<<square[9]<<"     |"<<e
ndl;
}
int checkwin()
{
        if(square[1]==square[2]&&square[2]==square[3])
                return 1;
        else if(square[4]==square[5]&&square[5]==square[6])
                return 1;
        else if(square[7]==square[8]&&square[8]==square[9])
                return 1;
        else if(square[1]==square[4]&&square[4]==square[7])
                return 1;
        else if(square[2]==square[5]&&square[5]==square[8])
                return 1;
        else if(square[3]==square[6]&&square[6]==square[9])
                return 1;
        else if(square[1]==square[5]&&square[5]==square[9])
                return 1;
        else if(square[3]==square[5]&&square[5]==square[7])
                return 1;
        else if(square[1]!='1'&&square[2]!='2'&&square[3]!='3'&&
                        square[4]!='4'&&square[5]!='5'&&square[6]!='6'&&
                        square[7]!='7'&&square[8]!='8'&&square[9]!='9')

                return 0;
        else return -1;
}
```

```
int main()
{       int choice,i;
        char mark;
        char a[10],b[10];
        int player=1;
        cout<<"\t***Welcome to tic toe Game***";
        cout<<"\nEnter player name= ";
        cin>>a>>b;
        cout<<"\n player 1= "<<a<<"\nplayer 2= "<<b;
        do
        {
                board();
                player=(player%2)?1:2;
                mark=(player==1)?'x':'o';
                cout<<"\nPlayer= "<<player;
                cout<<"\nEnter pos= ";
                cin>>choice;
                if(choice==1&&square[1]=='1')
                square[1]=mark;
                else if(choice==2&&square[2]=='2')
                square[2]=mark;
                else if(choice==3&&square[3]=='3')
                square[3]=mark;
                else if(choice==4&&square[4]=='4')
                square[4]=mark;
                else if(choice==5&&square[5]=='5')
                square[5]=mark;
                else if(choice==6&&square[6]=='6')
                square[6]=mark;
                else if(choice==7&&square[7]=='7')
                square[7]=mark;
                else if(choice==8&&square[8]=='8')
                square[8]=mark;
                else if(choice==9&&square[9]=='9')
    square[9]=mark;
                else
                {
                cout<<"***Invalid pos***";
                --player;
                cin.ignore();//In build function
                cin.get();//In build function gives one more chance
                }
                i=checkwin();
                player++;
}while(i==-1);
board();
if(i==1)
{
                --player;
                if(player--==1)
                {
```

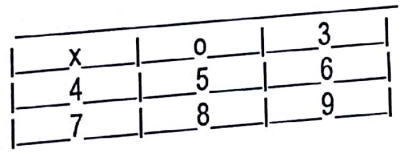
```
                                        TTT
                        cout<<"\n**** Player "<<a<<" win ;) *****";
                        cout<<"\n**** Player "<<b<<" loss :( *****";
                }
                else
                        {
                        cout<<"\n**** Player "<<b<<" win ;) *****";
                  cout<<"\n**** Player "<<a<<" loss :( *****";
                        }
                }
                //cout<<"\nplayer"<<--player<<" win";
        else
                cout<<endl<<"***Game Draw***";
    cin.ignore();
    cin.get();//Game again
    return 0;
}
/*
OUTPUT:-
        ***Welcome to tic toe Game***
Enter player name= 11
22
 player 1= 11
player 2= 22
```



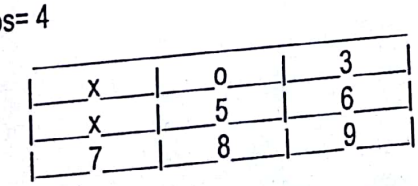

```
Player= 1
Enter pos= 1
```



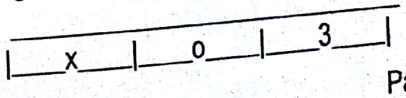

```
Player= 2
Enter pos= 2
```




```
Player= 1
Enter pos= 4
```




```
Player= 2
Enter pos= 5
```



TTT

x	o	6
7	8	9

Player= 1
Enter pos= 7

x	o	3
x	o	6
x	8	9

**** Player 11 win ;) ****
**** Player 22 loss :(****
*/