EUROPEAN UNIVERSITY OF LEFKE
Faculty of Engineering
Department of Computer Engineering

COMP218
OBJECT-ORIENTED PROGRAMMING

LAB WORK NO. 8

Prepared by **David O. Ladipo** (174574)
Submitted to Dr. Ferhun Yorgancıoğlu

Task-1: Consider below partly given class declaration.

a. Write definitions of the member functions listed above.

```
//default constructor
Vector::Vector(){
  size = 0;
}
//parameterized constructor
Vector::Vector(int c){
  size = c;
}
//copy constructor
Vector::Vector(const Vector& vec){

  size = vec.size;
}
//destructor
Vector::~Vector(){}
//set function
void Vector::setSize(int c){
  size = c;
}
//get function
int Vector::getSize()const{
  return size;
}
//overloaded "is equal to" operator
bool Vector::operator==(const Vector& vec)const{

  for(int i = 0;i < size;i++)
    if(list[i] != vec.list[i])
      return false;
  return true;
}
//overloaded "is not equal to" operator
bool Vector::operator!=(const Vector& vec)const{
  return !(*this == vec);
}
//overloaded "subscript" operator
int& Vector::operator[](int index){
  if(index < 0 || index > size - 1){
    cout<<"Out of bounds!!!" << endl;

  }
  return list[index];
}
//overloaded "subscript" operator
int Vector::operator[](const int index)const{
  if(index < 0 || index > size - 1){
     cout<<"Out of bounds!!!" << endl;
  }
  return list[index];
}
//overloaded "parathesis" operator
void Vector::operator()(int index,int data){
  if (index == size)
    push(data);
  else
    list[index] = data;
}
//overloaded "post-increment" operator
Vector Vector::operator++(){
  Vector t;
  this->size++;
  t.size = this->size;
  return t;
}
//overloaded "pre-increment" operator
Vector Vector::operator++(int){
  Vector t;
  t.size = this->size;
  this->size++;
  return t;
}
//I created a customized pushback function to test the driver program...
void Vector::push(int data)
{
    // if the number of elements is equal to the size,
    // that means we don't have space
    // to accommodate more elements.
    // We need to double the size
    if (current == size) {
      int newSize = 2 * size;
        int temp[newSize];

        // copying old array elements to new array
        for (int i = 0; i < size; i++) {
            temp[i] = list[i];
        }
```

```
              int list[newSize];
              for(int i=0;i < size;i++)
               list[i] = temp[i];
         }
         // Inserting data
         list[current] = data;
         current++;
     }

//overload “stream insertion” operator
    ostream& operator<<(ostream & display,const Vector& vec){

    for(int i = 0;i < vec.size;i++){
       if(i == vec.size - 1)
      display<<vec.list[i];
       else
      display<<vec.list[i]<<" ";
  }

          return display;
}
//overload “stream extraction” operator
     istream& operator>>(istream & input, Vector& vec){
        cout<<"Enter Element to push: ";
         int elem;
         input>>elem;
         vec.push(elem);
          return input;
}
```

b. Rewrite the program by separating the implementation file from the interface using a header file.

⇨Provide a driver program to test each implementation.

main.cpp

```
#include <iostream>
#include <stdlib.h>
#include "vector.h"

using namespace std;
```

```cpp
void menu(){
  cout << "=====Driver Program to test each Implementation=====" << endl;
  cout << "=====Using two Vectors, v1 and v2=====" << endl;
  cout << "============================================ " <<endl;
  cout << "1. Set size of Vector:v1" << endl;
  cout << "2. Enter Vector Elements: " << endl;
  cout << "3. Assign vector v1 to v2" << endl;
  cout << "4. Print vector v1 and v2" << endl;
  cout << "5. Check if vector v1 is is equal vector v2" << endl;
  cout << "6. Increment the size of vector v1 by 1" << endl;
  cout << "7. Print Vector v1 size" << endl;
  cout << "8. Select an Index to be printed" << endl;

}

int main() {

  int x, num, option;
  Vector v1,v2;

  menu();
  while(1){
    cout<<endl<<"[Choose any option from the MENU]"<<endl; cin>>option;

  switch(option){
    case 1:
    cout << "Enter size of vector" <<endl;
    cin >> x;
    v1.setSize(x);
    cout << " ============================" <<endl;
    break;

    case 2:
    cout << "Enter vector values: " << endl;
    for (int i = 0; i < x; i++)
    {
    cin >> num;
    v1.push(num);
    }
    cout << " ============================" <<endl;
    break;

    case 3:
    v2 = v1;
    cout << "Vector v1 has been assigned to v2 successfully"<<endl;
    cout << " ============================" <<endl;
    break;

    case 4:
    cout<<"Vector v1: " << v1 <<endl;
    cout<<"Vector v2: " << v2 <<endl;
    cout << " ============================" <<endl;
    break;

    case 5:
    if (v1 == v2){
    cout << "Vector v1 is equal to Vector v2" <<endl;
               }
    else
    cout << "They are not equal" << endl;
    cout << " ============================" <<endl;
    break;

    case 6:
    v1++;
    cout << "Vector v1 size has been incremented sucessfully" << endl;
    cout << " ============================" <<endl;
    break;

    case 7:
    cout << "Vector v1 size is: " << v1.getSize() << endl;
    cout << " ============================" <<endl;
    break;

    case 8:
    int a;
    cout << "Select an element i the vectore to be printed" <<endl;
    cout << "Enter index: " << endl;
    cin >> a;
    cout << v1[a];
    cout << " ============================" <<endl;
    break;

    default:
    cout<<"Option not available.. Please choose from the options above.."<<endl;
    break;


    }
```

```
  }

}
```

vector.cpp

```
#include <iostream>
#include <stdlib.h>
#include "vector.h"
#define MAX_SIZE 100

using namespace std;

    //default constructor
    Vector::Vector(){
      size = 0;
    }
    //parameterized constructor
    Vector::Vector(int c){
      size = c;
    }
    //copy constructor
    Vector::Vector(const Vector& vec){

      size = vec.size;
    }
    //destructor
    Vector::~Vector(){}
    //set function
    void Vector::setSize(int c){
      size = c;
    }
    //get function
    int Vector::getSize()const{
      return size;
    }
    //overloaded "is equal to" operator
    bool Vector::operator==(const Vector& vec)const{

      for(int i = 0;i < size;i++)
        if(list[i] != vec.list[i])
          return false;
      return true;
    }
```

```
//overloaded "is not equal to" operator
bool Vector::operator!=(const Vector& vec)const{
  return !(*this == vec);
}
//overloaded "subscript" operator
int& Vector::operator[](int index){
  if(index < 0 || index > size - 1){
    cout<<"Out of bounds!!!" << endl;

  }
  return list[index];
}
//overloaded "subscript" operator
int Vector::operator[](const int index)const{
  if(index < 0 || index > size - 1){
     cout<<"Out of bounds!!!" << endl;
  }
  return list[index];
}
//overloaded "parathesis" operator
void Vector::operator()(int index,int data){
  if (index == size)
    push(data);
  else
    list[index] = data;
}
//overloaded "post-increment" operator
Vector Vector::operator++(){
  Vector t;
  this->size++;
  t.size = this->size;
  return t;
}
//overloaded "pre-increment" operator
Vector Vector::operator++(int){
  Vector t;
  t.size = this->size;
  this->size++;
  return t;
}
//I created a customized pushback function to test the driver program...
void Vector::push(int data)
{
    // if the number of elements is equal to the size,
    // that means we don't have space
```

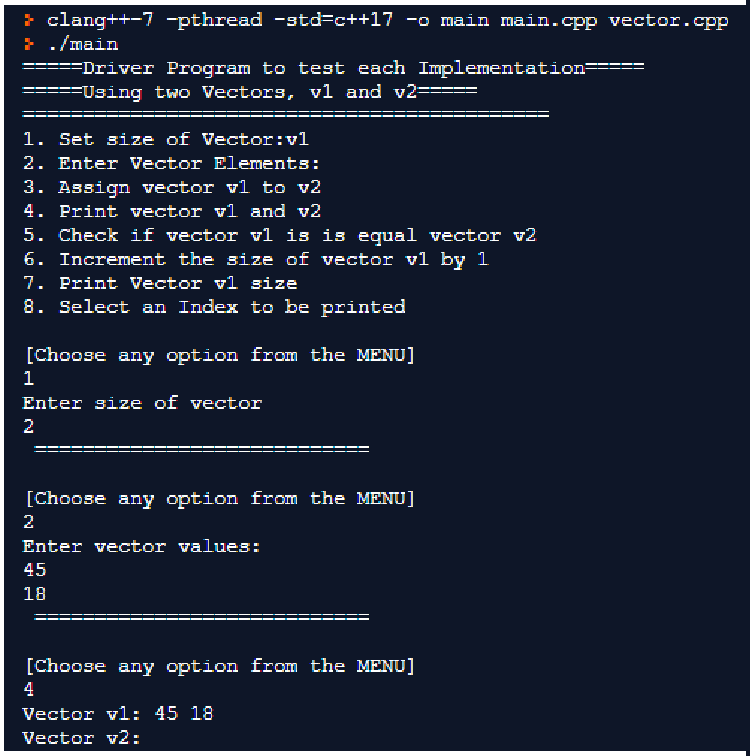
```
        // to accommodate more elements.
        // We need to double the size
        if (current == size) {
          int newSize = 2 * size;
            int temp[newSize];

            // copying old array elements to new array
            for (int i = 0; i < size; i++) {
                temp[i] = list[i];
            }
             int list[newSize];
             for(int i=0;i < size;i++)
              list[i] = temp[i];
        }
        // Inserting data
        list[current] = data;
        current++;
    }
    // function to delete last element

//overload “stream insertion” operator
    ostream& operator<<(ostream & display,const Vector& vec){

    for(int i = 0;i < vec.size;i++){
       if(i == vec.size - 1)
      display<<vec.list[i];
       else
      display<<vec.list[i]<<" ";
  }

         return display;
}
//overload “stream extraction” operator
     istream& operator>>(istream & input, Vector& vec){
        cout<<"Enter Element to push: ";
         int elem;
         input>>elem;
         vec.push(elem);
          return input;
}
```

vector.h

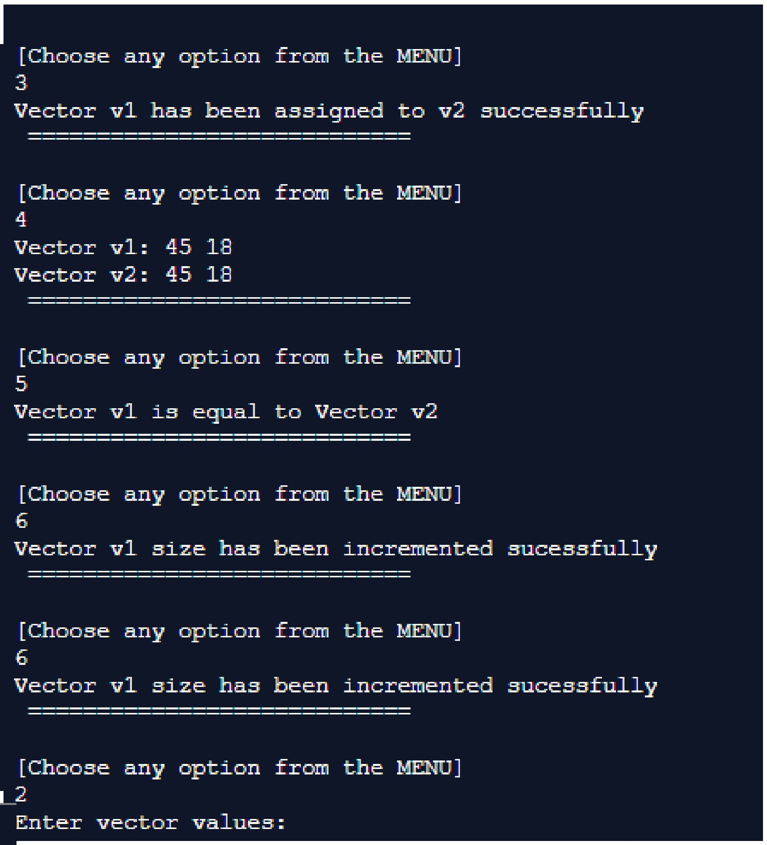
```
#ifndef VECTOR_H
```

```cpp
#define VECTOR_H
#define MAX_SIZE 100

using namespace std;

class Vector{
  public:
    //default constructor
    Vector();
    //parameterized constructor
    Vector(int);
    //copy constructor
    Vector(const Vector&);
    //destructor
    ~Vector();
    //Set function
    void setSize(int);
    //Get function
    int getSize()const;
    //overload “is equal” operator
    bool operator==(const Vector&)const;
    //overload “is not equal” operator
    bool operator!=(const Vector&)const;
    //overload “subscript” operator as a non-constant l-value
    int& operator[](int);
    //overload “subscript” operator as a constant r-value
    int operator[](const int)const;
    //overload “parenthesis” operator (passing index and value to be stored)
    void operator()(int,int);
    //overload “pre-increment” operator
    Vector operator++();
    //overload “post-increment” operator
    Vector operator++(int);
    //Created a customized push back function to test the driver program
    void push(int);
    //overload “stream insertion” operator
    friend ostream& operator<<(ostream &,const Vector&);
    //overload “stream extraction” operator
    friend istream& operator>>(istream &,Vector&);

  private:
    int list[MAX_SIZE];
    int size;
    int current; //the number of elements currently in the vector
};
```

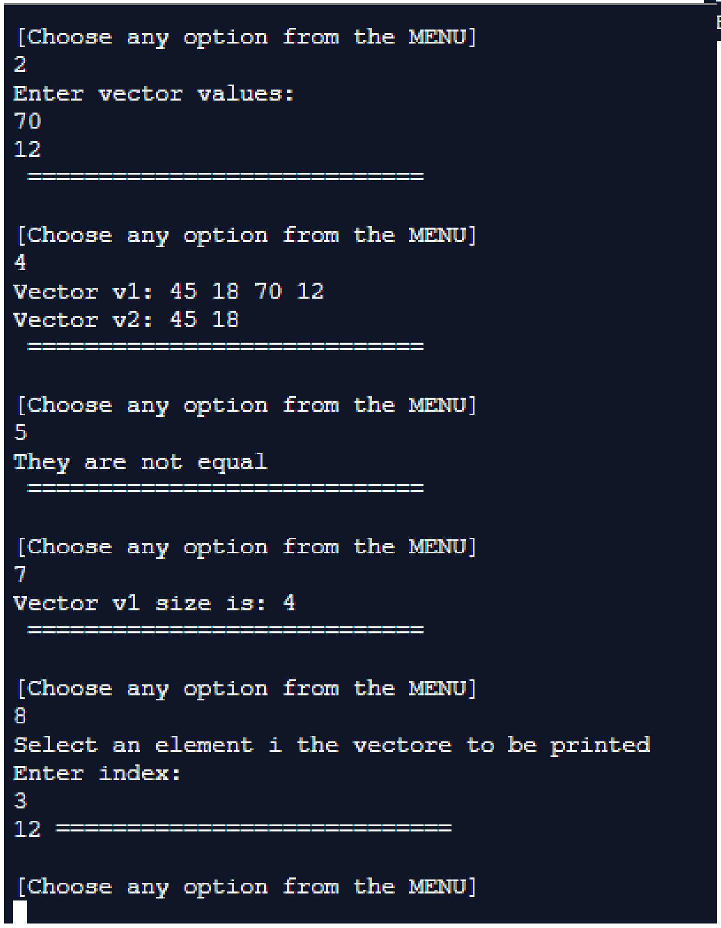
```
#endif
```



```
> clang++-7 -pthread -std=c++17 -o main main.cpp vector.cpp
> ./main
=====Driver Program to test each Implementation=====
=====Using two Vectors, v1 and v2=====
============================================
1. Set size of Vector:v1
2. Enter Vector Elements:
3. Assign vector v1 to v2
4. Print vector v1 and v2
5. Check if vector v1 is is equal vector v2
6. Increment the size of vector v1 by 1
7. Print Vector v1 size
8. Select an Index to be printed

[Choose any option from the MENU]
1
Enter size of vector
2
 ============================

[Choose any option from the MENU]
2
Enter vector values:
45
18
 ============================

[Choose any option from the MENU]
4
Vector v1: 45 18
Vector v2:
```



```
[Choose any option from the MENU]
3
Vector v1 has been assigned to v2 successfully
 ============================

[Choose any option from the MENU]
4
Vector v1: 45 18
Vector v2: 45 18
 ============================

[Choose any option from the MENU]
5
Vector v1 is equal to Vector v2
 ============================

[Choose any option from the MENU]
6
Vector v1 size has been incremented sucessfully
 ============================

[Choose any option from the MENU]
6
Vector v1 size has been incremented sucessfully
 ============================

[Choose any option from the MENU]
2
Enter vector values:
```



```
[Choose any option from the MENU]
2
Enter vector values:
70
12
 ============================

[Choose any option from the MENU]
4
Vector v1: 45 18 70 12
Vector v2: 45 18
 ============================

[Choose any option from the MENU]
5
They are not equal
 ============================

[Choose any option from the MENU]
7
Vector v1 size is: 4
 ============================

[Choose any option from the MENU]
8
Select an element i the vectore to be printed
Enter index:
3
12 ============================

[Choose any option from the MENU]
```

Task-2: Reconsider the Vector class declaration. Convert the implementation into a dynamic array form!

vector.h

```
#ifndef VECTOR_H
#define VECTOR_H

using namespace std;

class Vector{
  public:
    //default constructor
    Vector();
    //parameterized constructor
    Vector(int);
    //copy constructor
    Vector(const Vector&);
    //destructor
    ~Vector();
    //Set function
    void setSize(int);
    //Get function
    int getSize()const;
    //overload “is equal” operator
    bool operator==(const Vector&)const;
    //overload “is not equal” operator
    bool operator!=(const Vector&)const;
    //overload “subscript” operator as a non-constant l-value
    int& operator[](int);
    //overload “subscript” operator as a constant r-value
    int operator[](const int)const;
    //overload “parenthesis” operator (passing index and value to be stored)
    void operator()(int,int);
    //overload “pre-increment” operator
    Vector operator++();
    //overload “post-increment” operator
    Vector operator++(int);
    //Created a customized push back function to test the driver program
    void push(int);
    //overload “stream insertion” operator
    friend ostream& operator<<(ostream &,const Vector&);
    //overload “stream extraction” operator
    friend istream& operator>>(istream &,Vector&);
  private:
    int *list;
    int size;
    int current; //the number of elements currently in the vector
};
#endif
```

vector.cpp

```
#include <iostream>
#include <stdlib.h>
#include "vector.h"

using namespace std;

    // Default constructor to initialise
    // an initial capacity of 1 element and
    // allocating storage using dynamic allocation
    Vector::Vector(){
        list = new int[1];
        size = 1;
        current = 0;
    }
    void Vector::push(int data)
    {

        // if the number of elements is equal to the capacity,
        // that means we don't have space
        // to accommodate more elements.
        // We need to double the capacity
        if (current == size) {
            int* temp = new int[2 * size];

            // copying old array elements to new array
            for (int i = 0; i < size; i++) {
                temp[i] = list[i];
            }

            // deleting previous array
            delete[] list;
            size *= 2;
            list = temp;
        }

        // Inserting data
        list[current] = data;
        current++;
    }
    Vector::Vector(int c){
      size = c;
    }
    //copy constructor
    Vector::Vector(const Vector& vec){

      size = vec.size;
    }
    //destructor
    Vector::~Vector(){}
    //set function
    void Vector::setSize(int c){
      size = c;
    }
    //get function
    int Vector::getSize()const{
      return size;
    }
    //overloaded "is equal to" operator
    bool Vector::operator==(const Vector& vec)const{

      for(int i = 0;i < size;i++)
        if(list[i] != vec.list[i])
          return false;
      return true;
    }
    //overloaded "is not equal to" operator
    bool Vector::operator!=(const Vector& vec)const{
      return !(*this == vec);
    }
    //overloaded "subscript" operator
    int& Vector::operator[](int index){
      if(index < 0 || index > size - 1){
        cout<<"Out of bounds!!!" << endl;

      }
      return list[index];
    }
    //overloaded "subscript" operator
```

```
    int Vector::operator[](const int index)const{
      if(index < 0 || index > size - 1){
         cout<<"Out of bounds!!!" << endl;
      }
      return list[index];
    }
    //overloaded "parathesis" operator
    void Vector::operator()(int index,int data){
      if (index == size)
        push(data);
      else
        list[index] = data;
    }
    //overloaded "post-increment" operator
    Vector Vector::operator++(){
      Vector t;
      this->size++;
      t.size = this->size;
      return t;
    }
    //overloaded "pre-increment" operator
    Vector Vector::operator++(int){
      Vector t;
      t.size = this->size;
      this->size++;
      return t;
    }

//overload “stream insertion” operator
    ostream& operator<<(ostream & display,const Vector& vec){

    for(int i = 0;i < vec.size;i++){
       if(i == vec.size - 1)
      display<<vec.list[i];
       else
      display<<vec.list[i]<<" ";
  }

         return display;
}
//overload “stream extraction” operator
     istream& operator>>(istream & input, Vector& vec){
        cout<<"Enter Element to push: ";
         int elem;
         input>>elem;
```

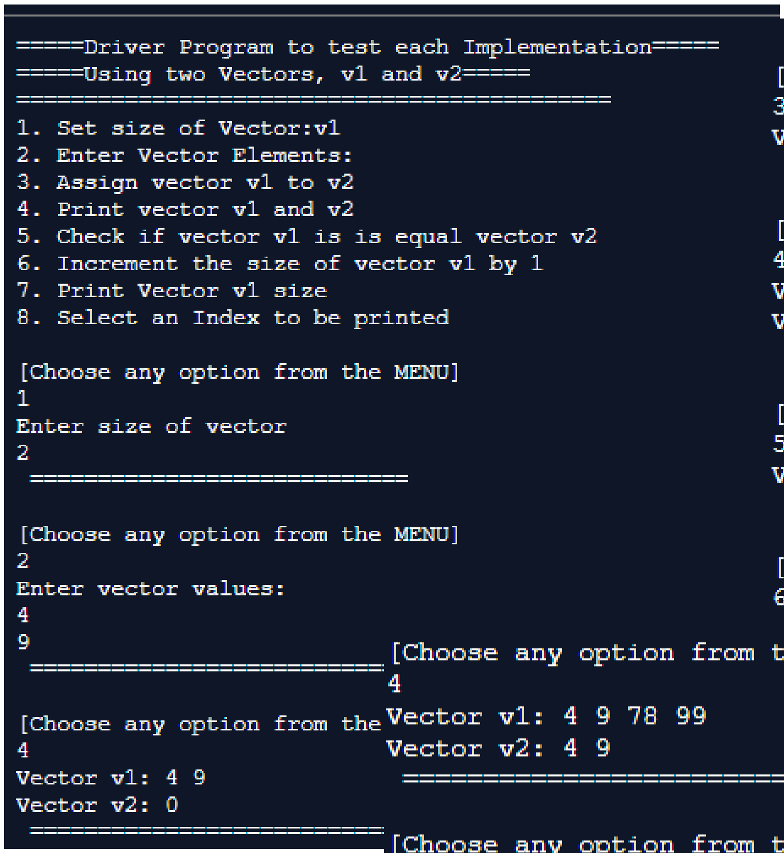
```
          vec.push(elem);
           return input;
}
```

main.cpp

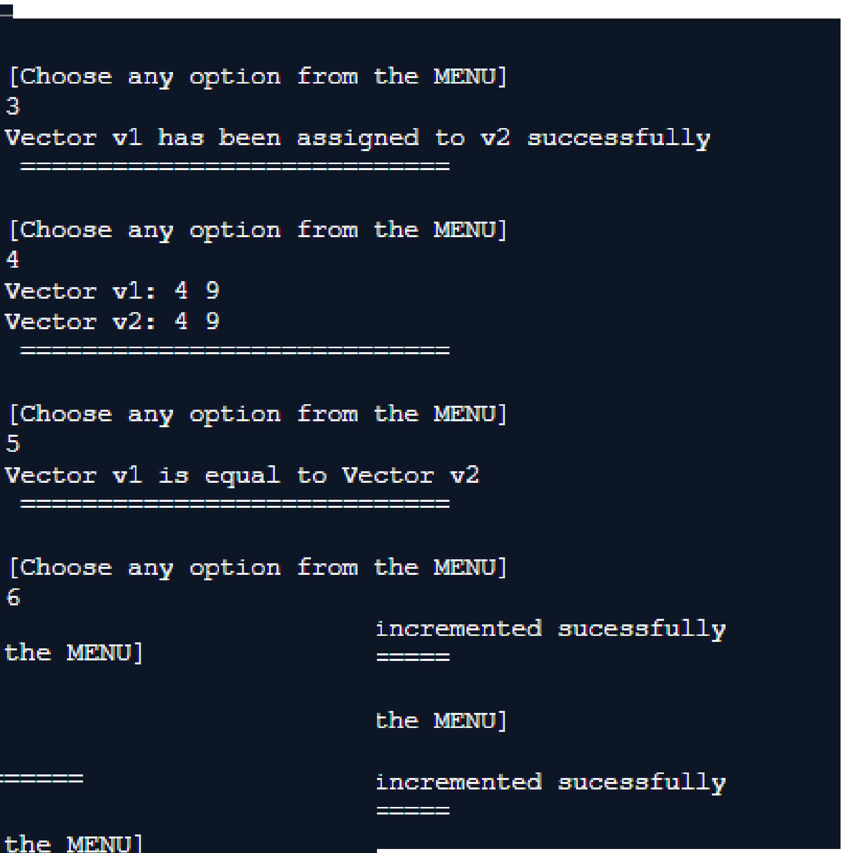
```
#include <iostream>
#include <stdlib.h>
#include "vector.h"

using namespace std;

void menu(){
  cout << "=====Driver Program to test each Implementation=====" << endl;
  cout << "=====Using two Vectors, v1 and v2=====" << endl;
  cout << "============================================ " <<endl;
  cout << "1. Set size of Vector:v1" << endl;
  cout << "2. Enter Vector Elements: " << endl;
  cout << "3. Assign vector v1 to v2" << endl;
  cout << "4. Print vector v1 and v2" << endl;
  cout << "5. Check if vector v1 is is equal vector v2" << endl;
  cout << "6. Increment the size of vector v1 by 1" << endl;
  cout << "7. Print Vector v1 size" << endl;
  cout << "8. Select an Index to be printed" << endl;

}

int main() {

  int x, num, option;
  Vector v1,v2;

  menu();
  while(1){
    cout<<endl<<"[Choose any option from the MENU]"<<endl; cin>>option;

  switch(option){
    case 1:
    cout << "Enter size of vector" <<endl;
    cin >> x;
    v1.setSize(x);
    cout << " ============================" <<endl;
    break;
```

```
case 2:
cout << "Enter vector values: " << endl;
for (int i = 0; i < x; i++)
{
cin >> num;
v1.push(num);
}
cout << " ============================" <<endl;
break;

case 3:
v2 = v1;
cout << "Vector v1 has been assigned to v2 successfully"<<endl;
cout << " ============================" <<endl;
break;

case 4:
cout<<"Vector v1: " << v1 <<endl;
cout<<"Vector v2: " << v2 <<endl;
cout << " ============================" <<endl;
break;

case 5:
if (v1 == v2){
cout << "Vector v1 is equal to Vector v2" <<endl;
           }
else
cout << "They are not equal" << endl;
cout << " ============================" <<endl;
break;

case 6:
v1++;
cout << "Vector v1 size has been incremented sucessfully" << endl;
cout << " ============================" <<endl;
break;

case 7:
cout << "Vector v1 size is: " << v1.getSize() << endl;
cout << " ============================" <<endl;
break;

case 8:
int a;
```

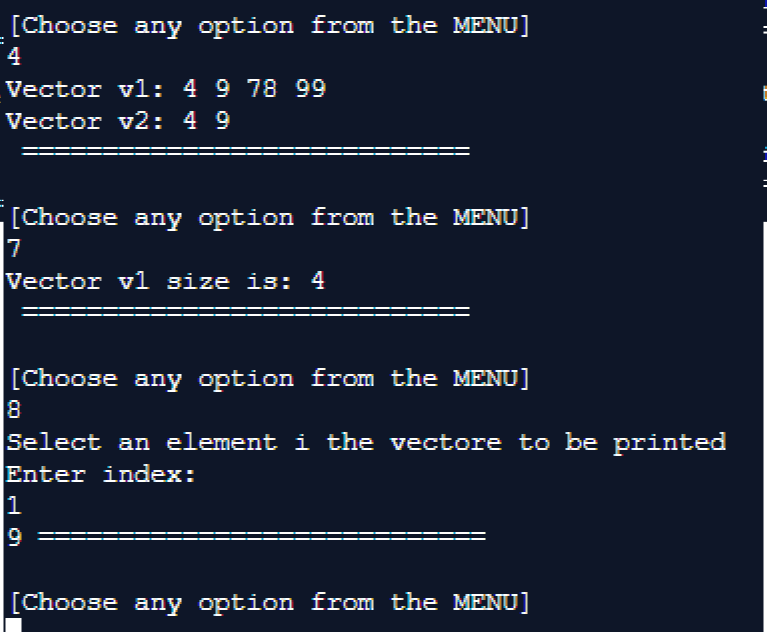
```cpp
        cout << "Select an element i the vectore to be printed" <<endl;
        cout << "Enter index: " << endl;
        cin >> a;
        cout << v1[a];
        cout << " ============================" <<endl;
        break;

        default:
        cout<<"Option not available.. Please choose from the options above.."<<endl;
        break;


        }
      }

    }
```



```
=====Driver Program to test each Implementation=====
=====Using two Vectors, v1 and v2=====
==========================================
1. Set size of Vector:v1
2. Enter Vector Elements:
3. Assign vector v1 to v2
4. Print vector v1 and v2
5. Check if vector v1 is is equal vector v2
6. Increment the size of vector v1 by 1
7. Print Vector v1 size
8. Select an Index to be printed

[Choose any option from the MENU]
1
Enter size of vector
2
 ============================

[Choose any option from the MENU]
2
Enter vector values:
4
9
 ==========================

[Choose any option from the
4
Vector v1: 4 9
Vector v2: 0
 ==========================
```



```
[Choose any option from the MENU]
3
Vector v1 has been assigned to v2 successfully
 ============================

[Choose any option from the MENU]
4
Vector v1: 4 9
Vector v2: 4 9
 ============================

[Choose any option from the MENU]
5
Vector v1 is equal to Vector v2
 ============================

[Choose any option from the MENU]
6
                        incremented sucessfully
                        =====

                        the MENU]

                        incremented sucessfully
                        =====
```



```
[Choose any option from the MENU]
4
Vector v1: 4 9 78 99
Vector v2: 4 9
 ============================

[Choose any option from the MENU]
7
Vector v1 size is: 4
 ============================

[Choose any option from the MENU]
8
Select an element i the vectore to be printed
Enter index:
1
9 ============================

[Choose any option from the MENU]
```