Name: Haseeb Ullah.

ID: F20232661009.

**Question 1:**

Create two classes DM and DB that store the value of distances. DM stores distance in

meters and centimeters and DB in feet and inches.

Write a program that can read values for the class objects and add one object of DM

with another object of DB. Use a friend function to carry out the addition operation. The

object that stores the results maybe a DM object or DB object, depending on the units

in which the results are required. The display should be in the format of feet and inches

or meters and centimeters depending on the object on display.

Hint:

 Feet = meters\*3.28084

 Inches = centimeter/2.54

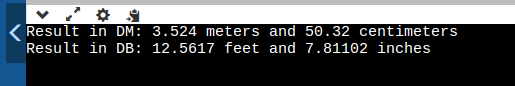
 1 meter = feet \* 0.3048

 Centimeter = inches \*2.54

Source Code;

|  |
| --- |
| #include <iostream>  using namespace std;  class DB;  class DM  {  double meters;  double centimeters;  public:  DM() : meters(0), centimeters(0) {}  DM(double m, double cm) : meters(m), centimeters(cm) {}  void setDistance(double m, double cm)  {  meters = m;  centimeters = cm;  }  void display()  {  cout << meters << " meters and " << centimeters << " centimeters" << endl;  }  friend DM add(DM, DB);  friend DB add(DB, DM);  };  class DB  {  double feet;  double inches;  public:  DB() : feet(0), inches(0) {}  DB(double ft, double in) : feet(ft), inches(in) {}  void setDistance(double ft, double in)  {  feet = ft;  inches = in;  }  void display()  {  cout << feet << " feet and " << inches << " inches" << endl;  }  friend DM add(DM, DB);  friend DB add(DB, DM);  };  DM add(DM d1, DB d2)  {  double totalMeters = d1.meters + d2.feet \* 0.3048;  double totalCentimeters = d1.centimeters + d2.inches \* 2.54;  while (totalCentimeters >= 100)  {  totalMeters += 1;  totalCentimeters -= 100;  }  return DM(totalMeters, totalCentimeters);  }  DB add(DB d1, DM d2)  {  double totalFeet = d1.feet + d2.meters \* 3.28084;  double totalInches = d1.inches + d2.centimeters / 2.54;  while (totalInches >= 12)  {  totalFeet += 1;  totalInches -= 12;  }  return DB(totalFeet, totalInches);  }  int main()  {  DM dm;  DB db;  dm.setDistance(2, 30);  db.setDistance(5, 8);  DM resultInDM = add(dm, db);  DB resultInDB = add(db, dm);  cout << "Result in DM";  resultInDM.display();  cout << "Result in DB: ";  resultInDB.display();  return 0;  } |

Output:



**Question 2:**

Write a program with a class integer that contains an array of integers. Initialize the

integer array in the constructor of the class. Then create friend functions to the class

Find the largest integer in the array.

Find the smallest integer in the array.

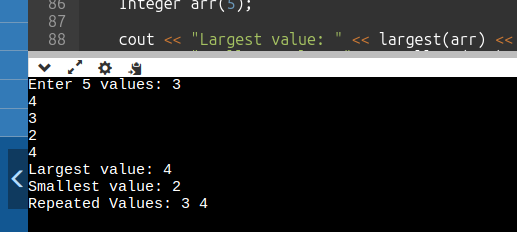
Find the repeated elements in array.

Create a destructor that sets all of the elements in the array to 0.

**Source Code:**

|  |
| --- |
| **#include <iostream>**  **using namespace std;**  **class Integer**  **{**  **int \*arr;**  **int size;**  **public:**  **Integer(int s) : size(s)**  **{**  **arr = new int[size];**  **cout << "Enter " << size << " values: ";**  **for (int i = 0; i < size; ++i)**  **{**  **cin >> arr[i];**  **}**  **}**  **~Integer()**  **{**  **for (int i = 0; i < size; ++i)**  **{**  **arr[i] = 0;**  **}**  **delete[] arr;**  **}**  **friend int largest(const Integer &);**  **friend int smallest(const Integer &);**  **friend void repeated(const Integer &);**  **};**  **int largest(const Integer &obj)**  **{**  **int largest = obj.arr[0];**  **for (int i = 1; i < obj.size; ++i)**  **{**  **if (obj.arr[i] > largest)**  **{**  **largest = obj.arr[i];**  **}**  **}**  **return largest;**  **}**  **int smallest(const Integer &obj)**  **{**  **int smallest = obj.arr[0];**  **for (int i = 1; i < obj.size; ++i)**  **{**  **if (obj.arr[i] < smallest)**  **{**  **smallest = obj.arr[i];**  **}**  **}**  **return smallest;**  **}**  **void repeated(const Integer &obj)**  **{**  **bool any\_repeated = false;**  **cout << "Repeated Values: ";**  **for (int i = 0; i < obj.size; i++)**  **{**  **for (int j = i + 1; j < obj.size; j++)**  **{**  **if (obj.arr[i] == obj.arr[j])**  **{**  **any\_repeated = true;**  **cout << obj.arr[i] << " ";**  **break;**  **}**  **}**  **}**  **if (!any\_repeated)**  **{**  **cout << "None";**  **}**  **cout << endl;**  **}**  **int main()**  **{**  **Integer arr(5);**  **cout << "Largest value: " << largest(arr) << endl;**  **cout << "Smallest value: " << smallest(arr) << endl;**  **repeated(arr);**  **return 0;**  **}** |

**Output:**



**Question 3:**

Write a program to swap the values of private data members of classes names class\_1 and

class\_2 using friend keyword.Code:

**Source Code:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Class\_2;  class Class\_1  {  int value1;  public:  Class\_1(int v = 0) : value1(v) {}  void display()  {  cout << "Class 1 value: " << value1 << endl;  }  friend void swapValues(Class\_1 &c1, Class\_2 &c2);  };  class Class\_2  {  int value2;  public:  Class\_2(int v = 0) : value2(v) {}  void display() const  {  cout << "Class 2 value: " << value2 << endl;  }  friend void swapValues(Class\_1 &c1, Class\_2 &c2);  };  void swapValues(Class\_1 &c1, Class\_2 &c2)  {  int temp = c1.value1;  c1.value1 = c2.value2;  c2.value2 = temp;  }  int main()  {  Class\_1 obj1(10);  Class\_2 obj2(20);  cout << "Before swapping:" << endl;  obj1.display();  obj2.display();  swapValues(obj1, obj2);  cout << "After swapping:" << endl;  obj1.display();  obj2.display();  return 0;  } |

**Output:**

