Task 1

Class A

#ifndef ClassA\_cpp

#define ClassA\_cpp

class A

{

public:

int getValuea() const { return valuea; }

void setValuea(int x) { valuea = x; }

//add a default constructor

A()

{

valuea = 0;

}

A(const A & other)

{

valuea = other.getValuea();

}

private:

int valuea;

};

#endif;

Class B

#ifndef ClassB\_cpp

#define ClassB\_cpp

#include "ClassA.cpp"

class B : public A

{

public:

int getValueb() const { return valueb; }

void setValueb(int x) { valueb = x; }

B()

{

valuea = 0;

valueb = 0;

}

B(const B & other)

{

valuea = other.getValuea();

valueb = other.getValueb();

}

private:

int valueb;

};

#endif;

Main

#include <iostream>

#include "ClassA.cpp"

#include "ClassB.cpp"

using namespace std;

int main()

{

A testA;

B testB;

testA.setValuea(75);

testB.setValuea(30);

testB.setValueb(100);

cout << "testA has a valuea of " << testA.getValuea() << endl;

cout << "testB has a valuea of " << testB.getValuea() << endl;

cout << "testB has a valueb of " << testB.getValueb() << endl;

Return 0;

}

Task 2

#include <iostream>

#include <string>

using namespace std;

template<class itemType>

void BubbleSort(itemType values[], int numValues)

{

bool swapped;

do

{

swapped = false;

for (int i = 1; i < numValues; i++)

{

if (values[i - 1] > values[i])

{

itemType temp = values[i - 1];

values[i - 1] = values[i];

values[i] = temp;

swapped = true;

}

}

} while (swapped == true);

}

int main()

{

int intArray[] = { 43, 7, 10, 23, 38, 4, 19, 51, 66, 14 };

float floatArray[] = { 43.2, 7.1, 10.5, 3.9, 18.7, 4.2, 19.3, 5.7, 66.8, 14.4 };

string strArray[] = { "cat", "dog", "fish", "frog", "bird", "hamster", "gerbil", "monkey", "elephant", "giraffe" };

BubbleSort(intArray, 10);

BubbleSort(floatArray, 10);

BubbleSort(strArray, 10);

for (int i = 0; i < 10; i++)

{

cout << intArray[i] << " ";

}

cout << endl;

for (int i = 0; i < 10; i++)

{

cout << floatArray[i] << " ";

}

cout << endl;

for (int i = 0; i < 10; i++)

{

cout << strArray[i] << " ";

}

cout << endl;

return 0;

}