***NAME : Himanshu Dixit***

***ENROLL NO. : 21103262***

***BATCH : B10***

***Software Development Lab – II [15B17CI271]***

***Assignment Sheet***

***Week 10***

***Q1.*** *Predict the output of following program.*

*template <typename T>*

*void abcd(const T&x)*

*{ static int count = 0;*

*cout << "x = " << x << " count = " << count << endl;*

*++count;*

*return;*

*}*

*int main()*

*{*

*abcd<int> (1);*

*cout << endl;*

*abcd<int>(1);*

*cout << endl;*

*abcd<double>(1.1);*

*cout << endl;*

*return 0;*

*}*

***Solution :***

x = 1 count = 0

x = 1 count = 1

x = 1.1 count = 0

***Q2.*** *Predict the output of the following program.*

|  |  |
| --- | --- |
| *template <class T>*  *class Test*  *{*  *private:*  *T val;*  *public:*  *static int count;*  *Test() { count++; }* | *int main()*  *{*  *Test<int> a;*  *Test<int> b;*  *Test<double> c;*  *cout << Test<int>::count << endl; cout << Test<double>::count << endl;* |

|  |  |
| --- | --- |
| *};*  *template<class T>*  *int Test<T>::count = 0;* | *return 0;*  *}* |

***Solution :***

2

1

***Q3.*** *Create a template class calculator to perform addition, subtraction, multiplication and division of two numbers. Show the results for different datatypes.*

***Solution :***

#include <iostream>

using namespace std;

template <class C>

class Calculator

{

private:

C num1,num2;

public:

Calculator(C n1, C n2)

{

num1=n1;

num2=n2;

}

void display()

{

cout<<"Numbers are : "<<num1<<" and "<<num2<<endl;

cout<<"Addition is = "<<add()<<endl;

cout<<"Subtraction is = "<<sub()<<endl;

cout<<"Division is = "<<div()<<endl;

cout<<"Multiplication is = "<<mul()<<endl;

}

C add()

{

return num1+num2;

}

C sub()

{

return num1-num2;

}

C div()

{

return num1/num2;

}

C mul()

{

return num1\*num2;

}

};

int main()

{

Calculator<int> c(1,1);

Calculator<float> f(3.1,2.1);

Calculator<double> d(5.5321,3.444);

cout<<"Int Result : "<<endl<<endl;

c.display();

cout<<"\nfloat Result : "<<endl<<endl;

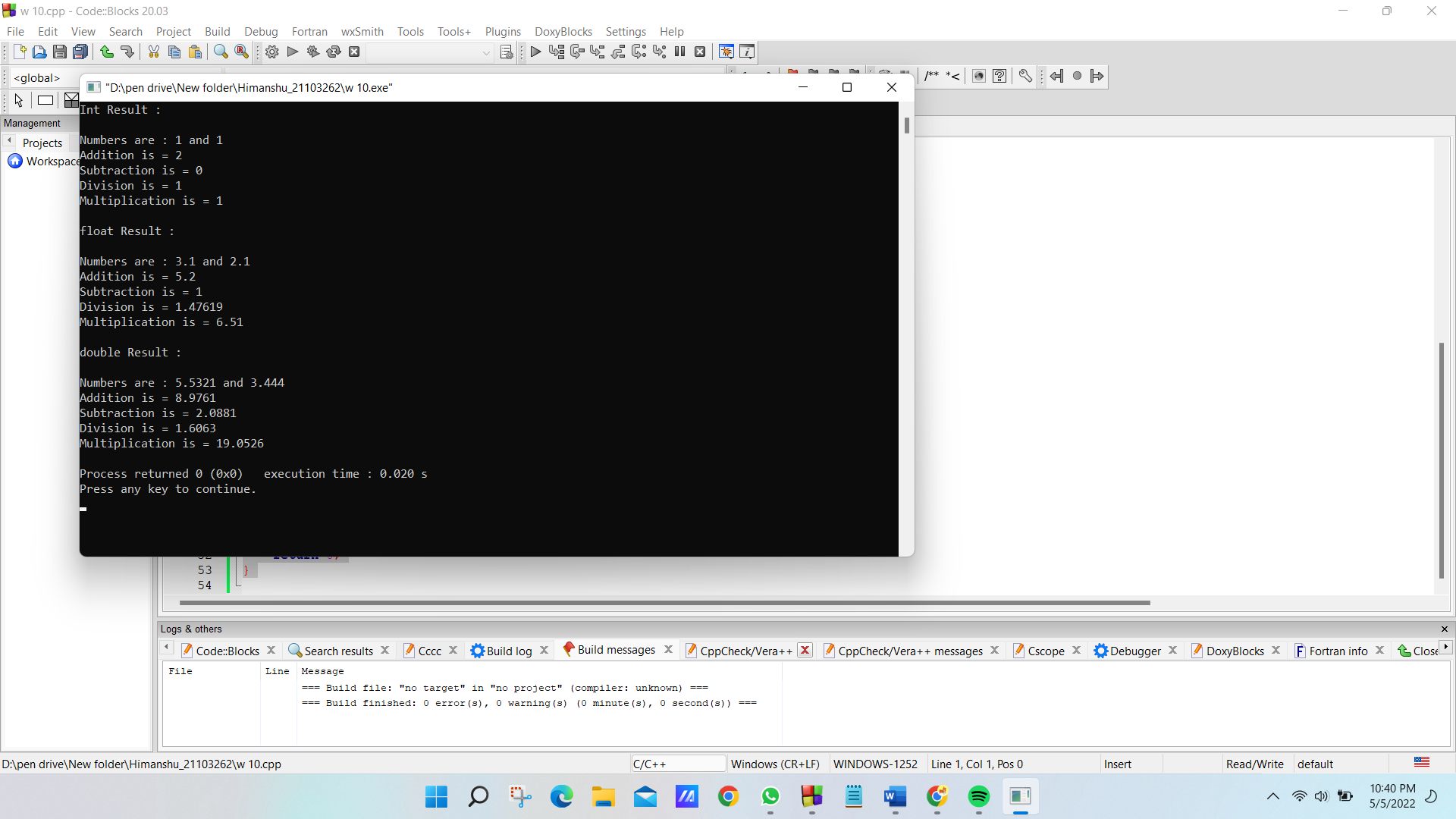
f.display();

cout<<"\ndouble Result : "<<endl<<endl;

d.display();

return 0;

}



***Q4.*** *Write templates for the two functions, namely minimum and maximum. Minimum function should accept two arguments and return the value of the arguments that is the lesser among the two. Maximum function should accept two arguments and return the value of the arguments that is the greater among the two values. Design a simple driver program that demonstrates the templates with various data types.*

***Solution :***

#include <iostream>

using namespace std;

template<class T>

T minimum(T a, T b)

{

return( (a>b)?b:a );

}

template<class T>

T maximum(T a, T b)

{

return( (a>b)?a:b );

}

int main()

{

cout<<"For integers values : 1 , 2\n\n";

cout<<"maximum is : "<<maximum(1,2)<<"\nminimum is : "<<minimum(1,2)<<endl;

cout<<"\nFor float values : 1.1 , 2.2\n\n";

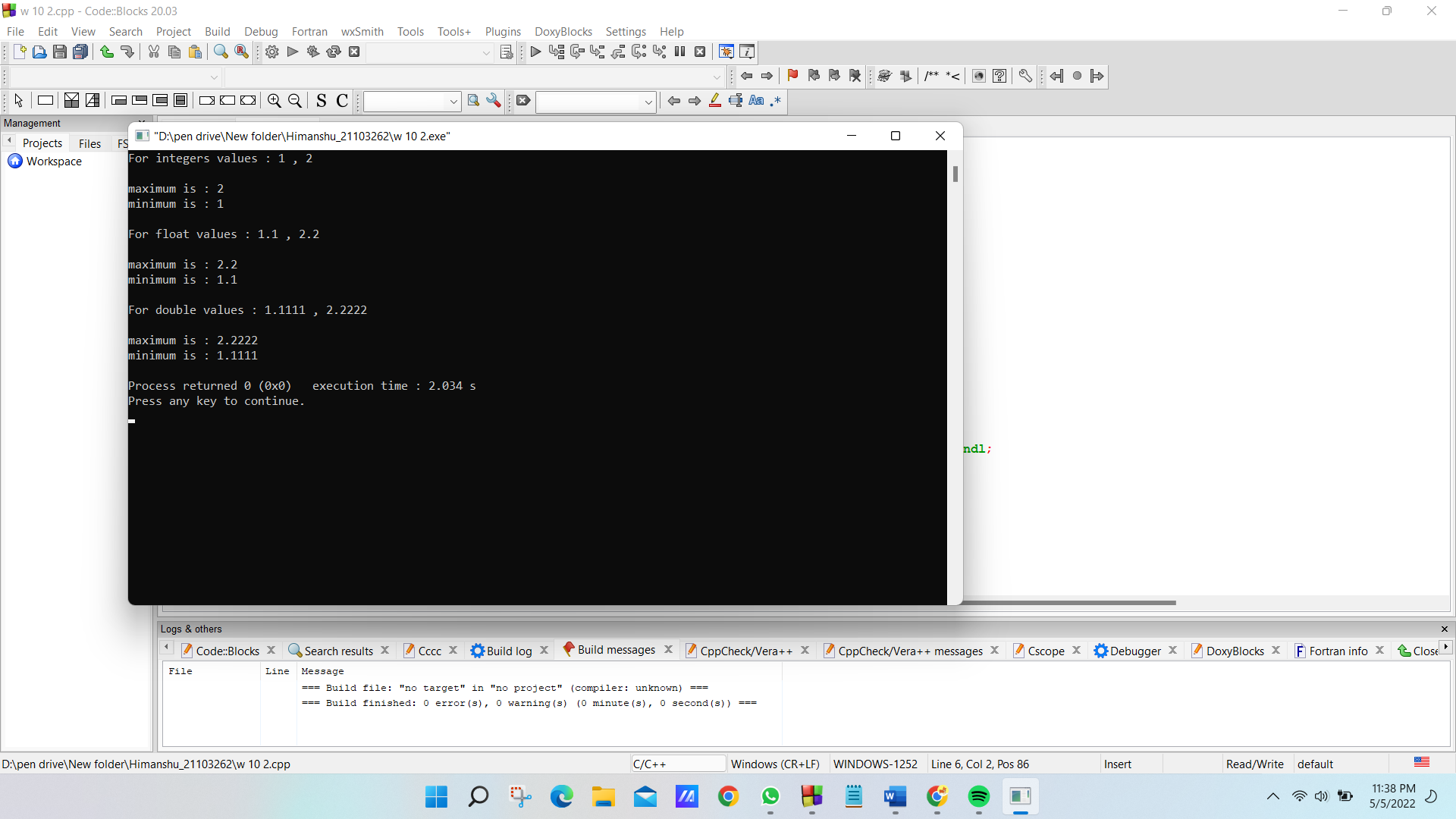
cout<<"maximum is : "<<maximum(1.1,2.2)<<"\nminimum is : "<<minimum(1.1,2.2)<<endl;

cout<<"\nFor double values : 1.1111 , 2.2222\n\n";

cout<<"maximum is : "<<maximum(1.1111,2.2222)<<"\nminimum is : "<<minimum(1.1111,2.2222)<<endl;

return 0;

}



***Q5. C****reate your own template class MyVector with data members and member functions such that the size(), push\_back() and pop\_back() functionalities of Vector can also be performed by MyVector.*

***Solution :***

#include <bits/stdc++.h>

using namespace std;

template <typename T> class vectorClass

{

T\* arr;

int current;

public:

vectorClass()

{

arr = new T[1];

current = 0;

}

void push(T data)

{

arr[current] = data;

current++;

}

void pop() { current--; }

int size() { return current; }

void print()

{

for (int i = 0; i < current; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

};

int main()

{

vectorClass<int> v;

v.push(10);

v.push(20);

v.push(30);

v.push(40);

v.push(50);

cout << "Vector size : " << v.size() << endl;

cout << "Vector elements : ";

v.print();

v.pop();

cout << "\nAfter deleting last element" << endl;

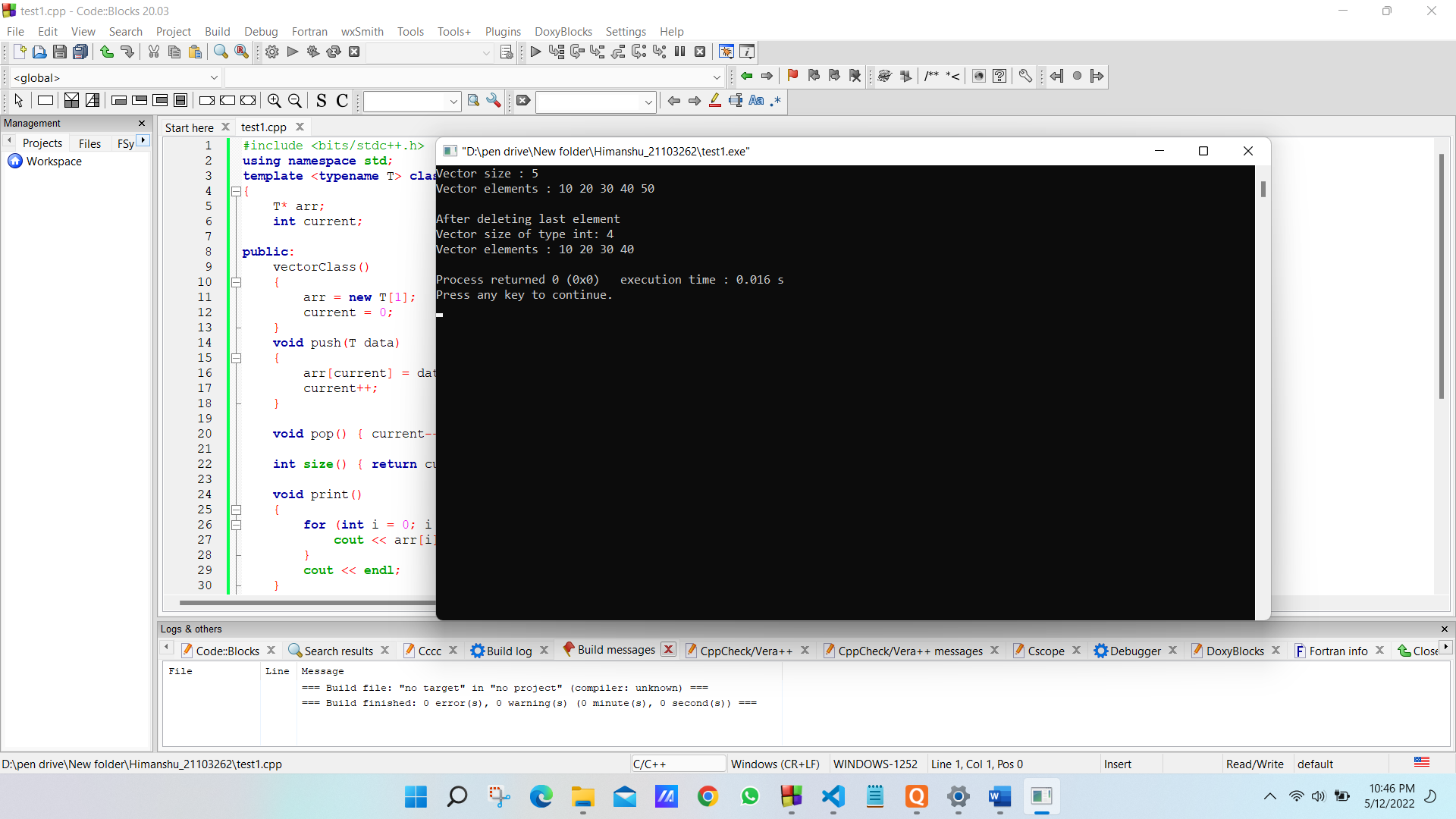
cout << "Vector size of type int: " << v.size() << endl;

cout << "Vector elements : ";

v.print();

return 0;

}



***Q6.*** *Implement the following problem using vector STL in c++.*

*Consider two arrays of similar type, having different numbers of elements. Take the array values from user till a negative value for both the arrays. Insert those elements of second array into first array which are not present in the first array. Also display the first array after insertion in sorted form.*

***Solution :***

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

vector<int> v;

vector<int>::iterator i;

while (true)

{

int a;

cout << "Enter the value:";

cin >> a;

if (a >= 0)

v.push\_back(a);

else

break;

}

sort(v.begin(), v.end());

for (i = v.begin(); i != v.end(); i++)

cout << \*i << " ";

return 0;

} 