Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Score:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ECE 2036 Final Exam**

Open book and notes, PCs and tablets allowed, but **no Internet Access** and C/C++ **code cannot be run** on a PC

1. (2%) The \_\_\_\_\_*list\_*\_\_\_\_ STL container is likely the most appropriate choice when new items are frequently inserted or deleted (not only at the ends) and random access is rarely if ever used.
2. (2%) An \_\_\_\_\_*iterator*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is normally used instead of a standard C++ pointer to move between data objects that are held inside STL containers.
3. (2%) Using inorder traversal, in what order would the nodes be processed in the following binary search tree.

Nodes processed in order = \_\_\_\_\_*DBEAC*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(just list letters)

 **A**

 / \

 **B C**

 / \

 **D E**

1. *(4%)* A C++ program needs to add new code to read and write an existing sequential file, *myfile.txt*. Write the C++ (not C-style) source statement in the space below that would open the file. Error checking should not be included and assume any necessary includes are already present.

*fstream myfileinout(“myfile.txt”, ios::in | ios::out );*

Assuming the file is open as described above, write a C++ source statement that would read an integer, *i*, from the file.

 *myfileinout >> i;*

1. (*10%*) Assuming the short C/C++ code segment below compiles, what is output?

 string str("ECE 2036 ");

 string mystr;

 str = str + "Final Exam";

 mystr = str.substr(6,5);

 cout << mystr << hex << mystr.length() << str[5] << str.length();

 cout.put(97).put(0x42);

 cout<< endl;

Output: \_\_\_\_\_\_\_\_\_\_\_\_*36 Fi5013aB*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (*4%*) What does the C/C++ code below print out?

*Enter mytry block*

*Got error #2036*

#include <iostream>

using namespace std;

int main()

{

 try

 {

 cout << "Enter mytry block" << endl;

 throw 2036;

cout << "End mytry block" << endl;

 }

 catch (int a) { cout << "Got error #" << a << endl; }

}

1. (*6%)* Write a C/C++ class template definition that automatically generates a new node for use in a doubly linked list. The class has the name, *DLLnode*. Each node in the list contains a single data value (of any type) stored as private data using the name *data*. Only a single argument constructor is required that supplies the initial value of the *data* in the node. The various get and set member functions should not be included (just to keep the answer shorter). Use static and not dynamic memory allocation to create the new node.

*template <typename NODETYPE>*

*class DLLnode*

*{*

*public:*

 *DLLnode::DLLnode (NODETYPE& info);*

*private:*

 *NODETYPE data;*

 *DLLnode <NODETYPE> \*forwardptr;*

 *DLLnode <NODETYPE> \*backptr;*

*};*

*template <typename NODETYPE>*

*DLLnode <NODETYPE>::DLLnode(NODETYPE &info)*

*: data(info), forwardptr(0), backptr(0)*

*{*

*}*

1. (*10%*) Are there any potential runtime problems with using the new class code shown below? If so, identify the main problem and add any additional C/C++ code needed to fix the code in the space provided in the code example.

Problem:\_\_\_\_\_\_\_\_\_\_*memory leak*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class BigArray

{

public:

 BigArray(int x);

 *~BigArray();*

private:

 int \*BAptr;

};

BigArray::BigArray(int x)

{

 BAptr = new int[x];

 }

*BigArray::~BigArray()*

*{*

 *delete [] BAptr;*

 *}*

1. (*10%*) What does the following C/C++ code output?

Answer:\_\_\_\_\_\_*06946*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <iostream>

using namespace std;

int main(int argc, \_TCHAR\* argv[])

{

 int a[10]={0,1,2,3,4,5,6,7,8,9};

 int \*aptr;

 aptr = &a[0];

 a[2] = a[2] + 3;

 a[3] = a[5] + a[4];

 aptr++;

 (\*aptr)--;

 (\*(++aptr))++;

 cout << a[1] << a[2] << a[3] << a[4] << \*aptr << endl;

}

1. *(10%)* Fill in the missing C/C++ code in the blank space provided below so that the expected output of “5 7 9” after adding two vectors with three elements will be produced by the main program’s computations and output by the first *cout*. Prototypes for the missing code are already included in the code. **Note:** Ignore the two comment lines with C/C++ code, they will be uncommented for use in the next problem.

#include <iostream>

using namespace std;

class Vector3D

{

// friend ostream &operator<<(ostream &, const Vector3D &);

public:

 Vector3D(int a, int b, int c);

 Vector3D operator+(const Vector3D& rhs) const;

 int x;

 int y;

 int z;

}; //***……………ADD THE MISSING C/C++ CODE BELOW………………………….***

*Vector3D::Vector3D(int a, int b, int c)*

*: x(a), y(b), z(c)*

*{*

*}*

*Vector3D Vector3D::operator+(const Vector3D& rhs) const*

*{*

 *Vector3D r(x+rhs.x,y+rhs.y,z+rhs.z);*

 *return r;*

*}*

int main()

{

 Vector3D V1(1, 2, 3);

 Vector3D V2(4, 5, 6);

 V1 = V1 + V2;

 cout << V1.x << " "<<V1.y <<" "<< V1.z << endl;

// cout << V1 << endl;

 return 0;

}

1. *(5%)* Assuming the code in the previous problem is added and works correctly, and that the two comment lines with C/C++ code are uncommented, write the additional C/C++ code needed in the space below so that “5 7 9” will also be generated by the *cout* *<<V1<<endl;*  line correctly.

*ostream &operator<<(ostream &output, const Vector3D &V)*

*{*

 *output << V.x << " "<< V.y <<" "<< V.z << endl;*

 *return output;*

*}*

1. *(15%)* Answer the following questions about the mbed C/C++ program attached at the end of the test. Assume that the hardware is connected as per the lab setups and that the missing includes and code at the beginning (not shown just for brevity here on final) have all been added so that the program compiles without errors and runs.
	1. List and describe the C++ object orientated language features that were used to setup the analog temperature sensor and make it relatively easy to use in this program?

*Class used for TMP36 – constructor sets up pin function for analog input in hardware*

*Float operator overload – reads and scales analog sensor value automatically*

* 1. What common hardware issue (i.e., not the need for external pullups) with pushbuttons was fixed by using the C/C++ code that activates the pushbutton callbacks?

*Pushbuttons need to be debounced and interrupts make it easier to sense pushbutton changes.*

* 1. Assuming that the external analog input pin used for the temperature sensor reads 1.4 volts with a voltmeter, what output would appear on the LCD when the program runs? *Hint:* the mbed analog temperature sensor conversion code is *return ((\_pin.read()\*3.3)-0.500)\*100.0*;

*90.00 F*

* 1. Describe the function of the two pushbuttons. What would a user notice when hitting the pushbuttons?

*pb1 adjusts speaker frequency*

*pb2 controls speaker volume*

*Mod (%) operator makes setting wrap around from highest back to lowest setting*

* 1. Is there anything else that a user would notice when running the program that is not dependent on hitting the pushbuttons?

*led1 on the mbed module changes state four times a second*

1. (*10%*) Write the output in the space below that is produced by the Constructor Destructor example C/C++ code provided with the test. Recall that most compilers use the copy constructor to make a new copy of the object whenever pass by value is used (instead of a pass by reference). Assume this also happens whenever a function returns a value that was previously setup as a local variable.



1. (*10%*) Write the output in the space below that is produced by the Inheritance and Polymorphism example C/C++ code provided with the test.



**C/C++ Code Examples for ECE 2036 Final Exam Problems**

//mbed code example for problem 12

//assume missing code and includes at beginning are present

//as per labs, and that the code below compiles and runs

//……

int volatile i=100;

int volatile j=5;

void pb1\_hit\_callback (void)

{

 i=(i+10)%500;

}

void pb2\_hit\_callback (void)

{

 j=(j+1)%10;

}

int main()

{

 float t=0.0;

 pb1.mode(PullUp);

 pb2.mode(PullUp);

 wait(.01);

 pb1.attach\_deasserted(&pb1\_hit\_callback);

 pb2.attach\_deasserted(&pb2\_hit\_callback);

 pb1.setSampleFrequency();

 pb2.setSampleFrequency();

 while (1) {

 t = myTMP36;

 printf("%5.2F",t);

 myLCD.cls();

 myLCD.printf("%5.2F C",t);

 mySpeaker.PlayNote(i\*10.0, 0.25, j/9.0);

 myLED1=!myLED1;

 }

}

// Constructors and Destructors Problem 13 Code

#include <iostream>

using namespace std;

class X{

 public:

 X(); // Default constructor

 X(int); // int Constructor

 X(const X&); // Copy constructor

 ~X(); // Destructor

 X operator+(const X& rhs) const; // Addition operator

 public:

 int x; // Single data member

};

X::X(): x(1){ cout<<"X Def Con "<<endl; };

X::X(int x): x(x){ cout<<"X Int Con "<<endl; };

X::~X(){ cout<<"-X Destructor "<<endl; };

X::X(const X &a){

 x=a.x;

cout<<"X Copy Con "<<endl;

};

 X X::operator+(const X& rhs) const

{

 cout<<"X + "<<endl;

 X r(x + rhs.x);

return r;

}

 class Y{

 public:

 Y(); // Default Constructor

 Y(int); // int Constructor

 Y(const Y&); // Copy constructor

 ~Y(); // Destructor

 Y operator+(Y rhs) const; // Addition operator

 public:

 int x; // Single data member

};

Y::Y(): x(1){ cout<< "Y Def Con "<<endl;};

Y::Y(int x): x(x){ x++; cout<< "Y Int Con "<<endl;};

Y::~Y(){ cout<< "-Y Destructor "<<endl;};

Y::Y(const Y &b){

 x=b.x;

 cout<<"Y Copy Con "<<endl;

};

 Y Y::operator+(Y rhs) const

{

 cout<< "Y + "<<endl;

 return Y(x + rhs.x);

}

int main()

{

 Y b(2);

 X a;

 Y c(b);

 b = b + c;

 a = a + a;

 cout << b.x << " " << c.x <<endl;

}

// Inheritance and Polymorphism problem 14 code

#include "stdafx.h"

#include <iostream>

using namespace std;

class Base

{ // Define a base class

public:

 virtual void Func1() = 0;

 virtual void Func2();

 virtual void Func3();

 void Func4();

};

class A : public Base

{ // Class A derives from Base

public:

 void Func2();

 virtual void Func4();

};

class B : public A

{// Class B derives from A

public:

 virtual void Func1();

 void Func2();

};

class C : public B

{ // Class C derives from B

public:

 virtual void Func1();

 virtual void Func4();

};

// Base Class Methods

void Base::Func2(){ cout << "Hello from Base::Func2()" << endl;}

void Base::Func3(){cout << "Hello from Base::Func3()" << endl;

Func1(); // DON’T MISS THIS CALL IN YOUR ANSWER

}

void Base::Func4(){ cout << "Hello from Base::Func4()" << endl;}

// Class A Methods

void A::Func2() { cout << "Hello from A::Func2()" << endl; }

void A::Func4() { cout << "Hello from A::Func4()" << endl; }

// Class B Methods

void B::Func1() { cout << "Hello from B::Func1()" << endl; }

void B::Func2() { cout << "Hello from B::Func2()" << endl; }

// Class C Methods

void C::Func1() { cout << "Hello from C::Func1()" << endl; }

void C::Func4() { cout << "Hello from C::Func4()" << endl; }

void TestFuncRef(Base& x)

{

 x.Func1();

 x.Func2();

 x.Func3();

 x.Func4();

}

void TestFuncVal(B x)

{

 x.Func1();

 x.Func2();

 x.Func3();

 x.Func4();

}

int main()

{

 B b;

 C c;

 TestFuncRef(c);

 TestFuncRef(b);

 TestFuncVal(c);

}