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

**ECE 2036 Test II**

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

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

string str("ECE 2036 ");

string mystr;

str = str + "Test 2";

mystr = str.substr(6,5);

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

cout.put(0x31);

cout<< endl;

Output: \_\_\_\_\_\_\_\_\_\_\_\_\_*36 Te50151*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (*5%*) What does the following code print out?

#include <iostream>

using namespace std;

int main()

{

try

{

cout << "Enter try block" << endl;

throw 2036;

cout << "End try block" << endl;

throw 2031;

}

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

}

|  |
| --- |
| *Enter try block* |
| *Error #2036* |
|  |

1. (*5%)* Write a C/C++ template definition that automatically generates a function called “AddEm” that returns the sum of two values of the same type (i.e., for integers AddEm(2, 3) returns 5, or for doubles AddEm(3.0, 2.0) returns 5.0). You can assume that the “+” operator is defined for any type that is used with the template.

*Template <class mytype>*

*mytype AddEM( mytype a, mytype b) {*

*return a+b;*

*}*

1. (5%) The *\_\_\_\_\_list\_\_\_\_\_\_\_* STL container is likely the most appropriate/efficient choice when new large items are frequently inserted or deleted (not only at the ends) and random access is never used.
2. (5%) An \_\_\_\_*\_iterator*\_\_\_\_\_\_\_\_\_\_\_ is normally used instead of a standard C++ pointer to move between data objects that are held inside STL containers.
3. (*10%*) If a new class definition needs dynamic memory allocation, the user also typically needs to

provide (i.e., not use the default) code for the classes’ \_\_\_\_*\_constructor*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, it creates the object using the C++ keyword \_\_*new\_\_\_\_\_\_\_\_\_\_.* The classes’ \_\_*destructor*\_\_\_\_\_\_\_\_ needs to reclaim the heap memory used by the object using the C++ keyword \_\_\_\_\_\_\_\_\_*delete*\_\_\_\_\_.

1. (*10%*) If you wrote a new class that was called “myclass”, write the additional statements needed to get a statement such as, “cout << A;” (where A is a myclass object) to automatically work in a program as one would expect. The data to print out with cout is private in myclass and setup using “int value;”.

Add this line to myclass definition to make it a friend:

*friend std::ostream& operator<<(ostream& output, const myclass rhs);*

*ostream& operator<<(ostream& output, const myclass rhs) {*

*output<<rhs.value;*

*return output;*

*}*

See textbook page 441

1. (*5%*) Why would using only the default constructor not work correctly in the matrix class from the matrix calculator lab and what could happen if it ran forever?

*Since dynamic memory allocation is used for a matrix (the size is not known at compile time) you must supply the constructor and destructor to avoid memory allocation errors and leaks.*

1. (*20% - 5% each part*) Answer the following questions about the mbed program attached at the end of the test. A speaker is connected to p21 and a pushbutton is connected to p14. The pushbutton reads a “0” until it is held down for a “1”. The program is a retro version of one of the very first computer games.
2. Explain what the display function does, using the value 0x0A as an illustrative example?

*It sends the low 4-bits of the value in binary to the four LEDs. With 0x0A the binary pattern seen in the four LEDs would be 1010 or ON OFF ON OFF.*

1. What would you notice watching the mbed module, assuming that the pushbutton is never hit?

*The pattern on the LEDs circular shifts to the left every .25 seconds.*

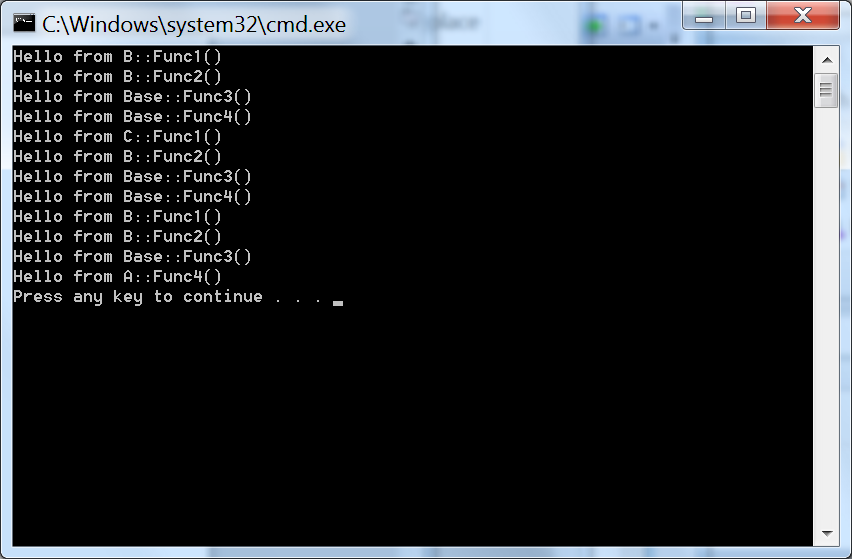
1. In the program, what condition on a variable is needed to make the speaker produce a sound, describe the sound that is made and what do the LEDs do when the speaker makes a sound?

*When value = 0 the speaker beeps a 2000Hz tone and LEDs flash on for .5 seconds, off for .25 seconds five times and then it restarts.*

1. Explain in simple terms the pushbutton strategy needed to win the game once it starts (i.e., produce a sound on the speaker)? Hint: Recall from 2030 that x XOR 0 = x and x XOR 1 = not x.

*Hit the pushbutton at the correct time to turn off all of the LEDs. Whenever the LED on the left is on, push down the pushbutton to kill it. If you miss and hit the button when the LED is on, another LED turns on.*

1. (*25%*) Write the output in the space below that is produced by the Inheritance and Polymorphism example C/C++ code provided with the test. You may have extra spaces in the table below.



**Mbed program used in problem 8**

DigitalOut myled1(LED1);

DigitalOut myled2(LED2);

DigitalOut myled3(LED3);

DigitalOut myled4(LED4);

DigitalIn pb(p14);

PwmOut spkr(p26);

void display(int number)

{

myled1 = (number) & 0x01;

myled2 = (number>>1) & 0x01;

myled3 = (number>>2) & 0x01;

myled4 = (number>>3) & 0x01;

}

int main()

{

unsigned int value = 0x12;

spkr.period(1.0/2000.0);

while(1) {

value = value ^ pb;

if (value == 0) {

for (int i=0; i<5; ++i) {

spkr = 0.5;

display(0x0F);

wait(.5);

display(0);

spkr = 0.0;

wait(.25);

}

value = 0x012;

}

value = ((value & 0x01)<<3) | (value >> 1); //circular shift

display(value);

wait(.25);

}

}

**// Problem 11 C++ code - Inheritance and Polymorphism**

#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 Func1();

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 Base

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;}

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

// Class A Methods

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

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(b);

TestFuncRef(c);

TestFuncVal(c);

}