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

**ECE 2036 Test 1**

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

1. (*5%)* Fill in the typical size (per textbook and slides) in bytes of the following C/C++ variable types:

int \_\_*4\_*\_\_ bytes char \_\_\_*1*\_\_\_bytes long long \_\_\_*8\_\_* bytes

double array[10] \_*80*\_*\_\_\_\_*\_\_ bytes C style string with 32 characters \_\_*33*\_\_\_\_ bytes

1. (*5%)* Where is the keyword “*explicit*” used in C/C++ and what does it do?

*It turns off automatic type conversion on function arguments*

1. (5%) The \_\_\_\_\_*Stack*\_\_\_\_\_\_\_\_\_\_\_\_\_ is the common name of the memory structure setup to store function return addresses and local variables. It also enables recursion.
2. (*5%)* Using the analog temperature sensor as in the mbed lab assignment, what C++ feature allowed the use of a simple statement like “*temp = mytemp*;” to read in the current temperature. *Temp* is type *float* and *mytemp* is a *TMP36* object.

*Operator overloading of float conversion of the TMP36 object*

1. (*10%)* Write a C/C++ *void* function called *clear* with one integer pointer argument that changes the value of the argument to 0 back in the calling program. Show an example call with the integer pointer variable *xptr*. Assume *xptr* is already declared as a pointer to an integer.

*void clear(int \*x);*

*{*

*\*x=0;*

*}*

*clear(xptr);//example call*

1. (*10%)* Write a C/C++ *for* statement using an integer loop control variable “*i*” that would add one to each “odd” element of an existing 1D integer array called “*array”. ”i”* must go out of scope when the *for* loop exits. There are 1000 elements in the array. It should skip the “even” *array* elements (i.e., array[0], array[2]…).

*for (int i=1; i<1000; i=i+2) {*

*array[i]=array[i]+1;*

*}*

1. (*16%)* Write the output produced by this C++ program exactly as it will appear on the output device.

*4 8 16*

*32 64 128*

*256 512*

**#include "stdafx.h" //used only on Windows OS**

**#include <iostream>**

**using namespace std;**

**class test**

**{**

**public:**

 **test(int x);**

 **void q(test y);**

 **int getw()**

 **{**

 **return w;**

 **}**

 **void setw(int y)**

 **{**

 **w = y;**

 **}**

**private:**

 **int w;**

**};**

**test::test(int x)**

**: w(0)**

**{**

 **setw(x);**

**}**

**void test::q(test y)**

**{**

 **w = y.w + getw() ;**

**}**

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

**{**

 **test A(2);**

 **int i = 1;**

 **while (i <= 8)**

 **{**

 **A.q(A);**

 **cout << A.getw() << " ";**

 **if (i%3==0) cout<< endl;**

 **i++;**

 **}**

**}**

1. (*18%)* Write the output that is produced by this C/C++ program. Assume array “a” starts at address 0x0030FDB8 and that the compiler prints out all 32-bit addresses in hexadecimal.

*0030FDB8 3511 0030FDBC 3*

**#include "stdafx.h" //needed only in Windows OS**

**#include <iostream>**

**using namespace std;**

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

**{**

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

 **int \*aptr;**

 **aptr = &a[1];**

 **a[2] = a[2] + 2;**

 **a[3] = a[2] + aptr[4];**

 **--aptr;**

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

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

 **cout <<&a<<" "<<a[1]<<a[2]<<\*(a+3)<<" "<<aptr<<" "<<aptr[0];**

**}**

1. (*14% total*) The following two questions refer to the mbed C++ program on the bottom of this page. A pushbutton is connected from P8 to gnd, and a jumper wire from P9 to gnd.

(*10%*) Describe exactly what this program does. Include pushbutton operation, LEDs, events, and time in your answer.

***Waits for pushbutton hit. The LEDs then count up in binary (0..15) twice per second. At 15 it resets back to 0.***

 (4%) What happens if the jumper wire is moved from P9 to Vout and mbed is reset?

***Counts down and not up***

**#include "mbed.h"**

**DigitalOut l1(LED1);**

**DigitalOut l2(LED2);**

**DigitalOut l3(LED3);**

**DigitalOut l4(LED4);**

**DigitalIn a(p8);**

**DigitalIn b(p9);**

**void doit( int a)**

**{**

 **l1 = a & 0x01;**

 **l2 = (a>>1) & 0x01;**

 **l3 = (a>>2) & 0x01;**

 **l4 = (a>>3) & 0x01;**

**}**

**int main()**

**{**

 **int c = 0;**

 **a.mode(PullUp);**

 **wait(.001);**

 **while(a!=0) { }**

 **while(1) {**

 **doit(c);**

 **if (b==0) c++;**

 **else c--;**

 **c = c % 16;**

 **wait(0.5);**

 **}**

**}**

1. (*12%)* Write the output in the space below that is produced by the Constructor Destructor example C/C++ code provided with the test on the last page. Recall that most compilers also 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. Note: There may be extra lines in the table provided below.

|  |
| --- |
| ***X Def Con*** |
| ***X Int Con*** |
| ***X Copy Con*** |
| ***X Inc*** |
| ***X Copy Con*** |
| ***X Copy Con*** |
| ***~X Destructor*** |
| ***~X Destructor*** |
| ***1 2 2*** |
| ***~X Destructor*** |
| ***~X Destructor*** |
| ***~X Destructor*** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**#include "stdafx.h" // needed only in Windows OS**

**// Constructors and Destructors Problem Code**

**#include <iostream>**

**using namespace std;**

**class X {**

**public:**

 **X(); // Default constructor**

 **X(int); // int Constructor**

 **X(const X&); // Copy constructor**

 **~X(); // Destructor**

 **X inc(X& y);**

**public:**

 **int x; // Single data member**

**};**

**X::X(): x(2){ 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::inc(X &b)**

**{**

 **cout <<"X Inc "<<endl;**

 **X y(b);**

 **y.x = y.x + 1;**

 **return y;**

**};**

**X c;**

**int main(){**

 **X a(1);**

 **X b(a);**

 **b = b.inc(b);**

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

**}**