The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, SPRING SEMESTER 2011-2012

G52CPP C++ Programming Examination

Time allowed TWO hours

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer Question ONE and TWO other questions

Marks available for sections of questions are shown in brackets in the right-hand margin.

Only silent, self-contained calculators with a single-line display are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

1 Compulsory Question, total 40 marks

(a) Consider the following lines of C++ source code. Say which lines, if any, would give compilation errors, and why. (6)

```
#include <iostream>
using namespace std;

struct S
{
    S( float f ) : f(f) {}
    float f;
};

class C
{
    C( float f ) : f(f) {}
    float f;
};

int main()
{
    S s1;
    C c1;
    S s2(1.0f);
    C c2(1.0f);

    cout << sizeof(S) << endl;
    cout << sizeof(C) << endl;

    return 0;
}
```

Problems:

1) **No default constructor on either, so the following two lines will fail:**

```
S s1;
C c1;
```

2) **The constructor in C is private, so the following line will also fail:**

```
C c2(1.0f);
```

(b) Provide C++ code to define an inline function called `multiply`, which takes two `double` parameters and returns the value of the first multiplied by the second, as a `double`. For example:

`multiply(3.1,2.0)` gives 6.2 and `multiply(1.1,-1.1)` gives -1.21 (4)

Straight from the notes really

(c) Convert your answer from part (b) into a template function which will take two parameters of the same type, apply the * (multiplication) operator to them and return a value of the same type. (6)

Straight from the notes really

(d) Consider the following source code. What would be the output if it was executed? (6)

```
char str[] = "Hello World\n";
char* p = str;

while ( *p++ )
    if ( *p != 'l' )
        cout << *p;
```

It will output all except the character 'l's (these are skipped by the 'if') and the initial H (the output comes after the p++)

(e) Consider the following source code. What would be the output if it was executed? (6)

```
int  i = 1;
int& j = i;
int  k = i;

for ( ; i < 10 ; i++, j++ )
{
    k = k + j;
    cout << i << "," << j << "," << k << endl;
}
```

Note the reference for j, so j and i are the same thing

(f) Consider the following source code. What would be the output if it was executed? (6)

```
#include <iostream>
using namespace std;

int foo( int i )
{
    static int j = i + 1;
    j = j + i;
    return j;
```

```
}

int main()
{
    int i = 1;
    int j = 1;
    do
    {
        j = foo(++i);
        cout << i << "," << j << endl;
    } while( i < 4 );

    return 0;
}
```

Here the trick is to note that ++I is a pre-increment – changing it before calling foo().
And the static j will be initialised ONCE, on the first call

(g) Consider the following source code. What would be the output if it was executed? (6)

```
#include <iostream>
using namespace std;

int i = 1;

int foo()
{
    int i = ++::i;
    i++;
    return i*i;
}

int main()
{
    i++;
    for ( int i = 0 ; i < 6 ; i++ )
    {
        {
            int i = foo();
            cout << i << endl;
        }
    }
```

```
        return 0;
    }
```

You need to know that the global i, the local variable in main and the local variable in foo() are separate. Consult the slides on visibility and scoping.
Note that the int i within the for loop is in a different block from the int i in the definition of the for loop, so will not affect the loop counter.

2 Optional question, total 30 marks (Answer Q1 and TWO other questions)

(a) What is the output of the following code? (12)

```
#include <iostream>
using namespace std;

class Base
{
public:
    virtual void foo() const { cout << "alpha" << endl; }
    void bar() const { cout << "beta" << endl; }
};

class Sub : public Base
{
public:
    void foo() const { cout << "gamma" << endl; }
    void bar() const { cout << "delta" << endl; }
};

class SubSub : public Sub
{
public:
    void foo() const { cout << "epsilon" << endl; }
    void bar() const { cout << "zeta" << endl; }
};

int main()
{
    Base b;
    Sub s;
    SubSub ss;
    Sub& r1 = ss;
    Base& r2 = ss;
    Base& r3 = s;
    Base* pBase[6] = { &ss, &s, &b, &r1, &r2, &r3 };

    for ( int i = 0 ; i < 6 ; i++ )
    {
        pBase[i]->foo();
        pBase[i]->bar();
    }
    return 0;
}
```

foo() is virtual, so the type of object it is on matters. bar() is not, so the type of pointer matters. The array is an array of base* pointers, so all calls through the array will use a base class pointer type

(b) Assume that the Base, Sub and SubSub classes are defined as in part (a). What is the output of the following code? (10)

```
class Data
{
    int* pi;
public:
    Data()
    {
        pi = new int[1024];
        for ( int i = 0 ; i < 16 ; i++ )
            pi[i] = i * 2;
    }

    int get( int i )
    { return pi[i]; }

    ~Data() { delete[] pi; }
};

void func( Data* pd, int i )
{
    if ( (i % 5) == 0 )
        throw pd->get(i);
    else if ( (i % 5) == 1 )
        throw Base();
    else if ( (i % 5) == 2 )
        throw Sub();
    else if ( (i % 5) == 3 )
        throw SubSub();
    cout << "Worked" << endl;
    cout << pd->get(i) << endl;
}

int main()
{
    for ( int i = 0 ; i < 10 ; i++ )
    {
        try
        {
            Data d;
            func( &d, i );
        }
```

```
        catch( const Sub& c )
        {
            cout << "Caught class Sub" << endl;
            c.foo();
        }
        catch( const SubSub& c )
        {
            cout << "Caught class SubSub" << endl;
            c.foo();
        }
        catch( const Base& c )
        {
            cout << "Caught class Base" << endl;
            c.foo();
        }
        catch( const Base* c )
        {
            cout << "Caught Base Pointer" << endl;
            c.foo();
        }
        catch( const int& i )
        {
            cout << "int exception caught" << endl;
            cout << i << endl;
        }
        catch( ... )
        {
            cout << "Unknown exception" << endl;
        }
    }
    return 0;
}
```

Remember that catch clauses are checked in the order that they appear, and that a sub-class object IS a base class object.

(c) Another programmer tells you that this program allocates more than 1k on the stack every time the Data object is created, and tells you that this is a bad idea. He suggests instead creating this object on the heap, by replacing the lines of code:

```
            Data d;
            func( &d, i );
```

by these lines:

```
            Data* pd = new Data();
            func( pd, i );
```

```
delete pd;
```

Briefly explain: Where is memory allocated (or objects created), is memory allocated from the stack or the heap and is it correctly deallocated? Is the programmer correct? Do you think that the proposed change is a good idea or a bad idea and what effect (or side-effects) do you expect this change would have? (8)

This uses an example of RAII, so if you understand that you should be able to do this. Look into the lecture slides for lecture 16, about RAII. As long as the objects are created on the stack they are destroyed correctly. Note that the memory for the big array is created on the heap, on the stack, using new, so it isn't a problem anyway.

3 Optional question, total 30 marks (Answer Q1 and TWO other questions)

(a) Consider the following source code for a class which will hold x and y coordinates:

```
class Coord
{
public:
    Coord() { }

    Coord( double x, double y)
    : x(x), y(y) { }

    ~Coord() { }

    double getX() { return x; }
    double getY() { return y; }
private:
    double x;
    double y;
};
```

Four methods can be considered to be created implicitly by the compiler if they are needed. A default constructor is one of these, and a destructor is another. Implementations of both of these are included in the code above.

(i) What are the two other functions which can be created implicitly? (4)

(ii) Provide one or more example lines of code which would force the compiler to generate the two functions in (i) above. State clearly which line(s) would force which of the functions to be generated. (4)

(iii) Provide an implementation of each of these two functions in your answer for (i), for the `Coord` class above. (6)

This is basically straight from the notes. A copy constructor and an assignment operator.

(b) Consider the following code which defines a class and declares a function:

```
class Position
{
    double x;
    double y;
public:
    Position( double x, double y)
        : x(x), y(y)
    { }
};

void doSomething( const Position& pos );
```

Note that the doSomething() function takes a single parameter, which is a Position object.

Since the Position and Coord classes are so similar, we would like to be able to pass a Coord object as a Position object, so that the following code would compile and execute correctly, without modification:

```
Coord c3( 1.0, 2.0 );  // Create a Coord object
doSomething( c3 );     // Pass it as a Position object
```

This can be achieved by adding one function to the Coord class. Provide the definition for a function which could be added to the Coord class to allow the above code to compile and execute correctly, without any modification. (5)

You are going FROM the Coord class to a Position class implicitly so you need a conversion operator in the Coord class.

```
operator Position ()
{
        return Position( x, y );
}
```

(c) Consider the following definition for the doSomething() function, which merely prints the position:

```
void doSomething( const Position& pos )
{
    // Do something with the position
    cout << pos.x << " , " << pos.y << endl;
}
```

Given the definition of the Position class in (b), the above implementation for the doSomething() function would not compile. Provide the line(s) of code which would need to be added to the Position class to allow the doSomething() function to compile and execute correctly, without modifying the doSomething() function and without making the x and y members of Position public. (4)

Function can easily be made a friend:
friend void doSomething(const Position& pos);

(d) It would be useful to be able to add two Coords objects together, to obtain a new Coord object. Provide an implementation of the + operator which will add the two coordinates to give a new Coord object. The new x coordinate should be the sum of the old coordinates, and the new y coordinate the sum of the y coordinates. The following code should then output the values 4.4 , 6.6:

```
Coord ca( 1.1, 2.2 );
Coord cb( 3.3, 4.4 );
Coord cc = ca + cb;
doSomething( cc );
```

Implement a + operator

```
Coord operator+( const Coord& rhs )
{
        return Coord( x + rhs.x, y + rhs.y );
}
```

4 Optional question, total 30 marks (Answer Q1 and TWO other questions)

(a) Consider the following source code:

```
#include <iostream>
using namespace std;

double d = 1.1;

class C
{
public:
    C( double d = 2.3 )
    : d(d)
    {
        d += 1.2;
    }

    void out() { cout << d << endl; }
private:
    double d;
};

void output1( const C& c )
{
    C* pc = XXXXXXXXXXXXXX<C*>(&c);
    pc->out();
}

void output2( const C& c )
{
    c.out();
}

int main()
{
    C c;
    output1( c );
    output2( c );
    return 0;
}
```

(i) What needs to be added in the space labelled XXXXXXXXXXXXXX in the `output1()` function to make it compile, assuming that the class C is not changed. (3)

const_cast

(ii) The `output2()`function would also not compile at the moment. What small change could be made to something in the class C to allow the `output2()` function to compile without having to modify the `output2()`function? (3)

Add 'const' specifier to the out function, which is valid since it is not a mutator.

(iii) Assuming that the modifications in (i) and (ii) are made, what is the output of this program? (3)

Note that the global d can be ignored, and the d+= 1.2 applies to the parameter, not the member.

(b) What is the output of the following code? (9)

```
#include <iostream>
using namespace std;

struct B1
{
    virtual void out1() { cout << "B1O1" << endl; }
};

struct B2
{
    virtual void out2() { cout << "B2O2" << endl; }
};

struct S1 : public B1, public B2
{
    virtual void out1() { cout << "S1O1" << endl; }
};

struct S2 : public B1, public B2
{
    virtual void out2() { cout << "S2O2" << endl; }
};

struct S3 : public S2
{
    virtual void out1() { cout << "S3O1" << endl; }
};

int main()
{
    S1 s1;
    S2 s2;
    S3 s3;

    B1* b1[] = { &s1, &s2, &s3 };
    B2* b2[] = { &s1, &s2, &s3 };

    for ( int i = 0 ; i < 3 ; i++ )
```

```
        b1[i]->out1();
    for ( int i = 0 ; i < 3 ; i++ )
        b2[i]->out2();

    return 0;
}
```

All functions are virtual so it is basically asking whether you know which type of object each is and whether each can cope with multiple base classes.

(c) What is the output of the following code? (12)

```
#include <iostream>

using namespace std;

typedef int (*ifi)(int);

int f1( int i ) { return 4; }
int f2( int i ) { return i/2; }
int f3( int i ) { return i+2; }

int main( int argc, char** argv )
{
    ifi a[] = {f1,f2,f3,f3,f2,f1,f2};
    int b[] = {4,2,10,12,8,6,4};

    for ( int i = 0 ; i < 6 ; i++ )
        cout << a[i](b[i]) << endl;

    for ( int i = 0 ; i < 6 ; i++ )
        cout << a[i](a[i+1](b[i])) << endl;

    return 0;
}
```

This is just some code using function pointers.

5 Optional question, total 30 marks (Answer Q1 and TWO other questions)

(a) What is the output of the following code? (10)

```
#include <iostream>
#include <vector>
using namespace std;

class Base
{
public:
    Base( int i = 42 )
    : i(i) { }

    virtual void out()
    { cout << "Base: " << i << endl; }
protected:
    int i;
};

class Sub : public Base
{
public:
    Sub( int i = 21, int j = 53 )
    : j(j), Base(i) { }

    virtual void out()
    { cout << "Sub: " << i << ", " << j << endl; }
protected:
    int j;
};

int main()
{
    Base* ap[5];
    ap[0] = new Sub( );
    ap[1] = new Sub( 3 );
    ap[2] = new Sub( 5, 6 );
    ap[3] = new Base();
    ap[4] = new Base( 8 );

    for( int i=0 ; i < 5 ; i++ )
        ap[i]->out();

    vector<Base> v(5);
    for( int i=0 ; i < 5 ; i++ )
        v[i] = *ap[i];
    for( int i=0 ; i < 5 ; i++ )
        v[i].out();

    return 0;
}
```

First 5 values is whether you understand default values. Next five illustrate the slicing problem.

(b) Consider the following code, which should maintain a list of strings?

```
#include <iostream>
#include <string>
using namespace std;

struct ListItem
{
    ListItem* pNext;
    string str;
};

void add( ListItem** ppFirst, string str )
{
    ListItem* pNew = new ListItem;
    pNew AAAAA *ppFirst;   // Part i
    pNew BBBBB str;        // Part ii
    *ppFirst = pNew;
}

int main()
{
    ListItem* pFirst = NULL;
    add( &pFirst, "One" );
    add( &pFirst, "Two" );
    add( &pFirst, "Three" );

    return 0;
}
```

(i) What should replace the text AAAAA in the code? (2)

->pNext =

(ii) What should replace the text BBBBB in the code? (2)

->str =

(iii) Provide an implementation for a function:

```
void output1( ListItem* pFirst )
```

which will output the contents of the list in the order they are stored. i.e. the item pointed at by `pFirst` should be output first. (6)

Example:

```
void output1( ListItem* pThis )
{
        while ( pThis != NULL )
        {
                cout << pThis->str << endl;
                pThis=pThis->pNext;
```

```
        }
}
```

(iv) Provide an implementation for a function:

`void output2( ListItem* pFirst )`

which will output the contents of the list in reverse order, so that the item pointed at by `pFirst` should be output last. (7)

Recursion is the easiest way to do this:

```
void output2( ListItem* pThis )
{
        if ( pThis == NULL )
                return;
        output2( pThis->pNext );
        cout << pThis->str << endl;
}
```

(v) What is the output of the above program, if a call to `output1(pFirst)` (as implemented by you in (iii)) is added immediately before the '`return 0;`' in the main() function. (3)

Note that it will store them in reverse order!