The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, SPRING SEMESTER 2010-2011

C/C++ for Java Programmers
(Module G52CFJ)

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, 40 marks

(a) Consider the following lines of C source code.

```
#include <stdio.h>

int main( int argc, char* argv[] )
{
    char* s1 = "C-code";
    char* s2 = "strings";
    char s3[] = { 'C', '-', 'c', 'o', 'd', 'e' };
    char s4[] = "C-code";

    return 0;
}
```

Assume that the code is compiled using a standard C compiler and that the resulting program executes without crashing.

State which of the following are guaranteed to be true, regardless of the size of pointers or the layout of local variables in memory.

(i) `strlen(s1) == 6`

(ii) `sizeof(s1) == 6`

(iii) `strlen(s2) == 7`

(iv) `sizeof(s2) == 7`

(v) `strlen(s3) == 6`

(vi) `sizeof(s3) == 6`

(vii) `strlen(s4) == 6`

(viii) `sizeof(s4) == 6`

(ix) `sizeof(s2) == sizeof( char* )`

(x) `sizeof(s4) == sizeof( char* )`

(10)

(b) Provide C/C++ code to define a macro (using `#define`) called `PRODUCT`, which takes two parameters and returns the product of the two parameters.

e.g. `PRODUCT(2,3)` is 6, `PRODUCT(1,5)` is 5 and `PRODUCT(-3,4)` is -12 (6)

(c) Consider the following source code:

```
int main( int argc, char* argv[] )
{
    char arr1[] = "Array";         // A
    char arr2[] = "Array";         // B
    const char * cp = arr2;        // C

    cp[0] = 'S';                   // D
    cp = arr2;                     // E

    char* p = XXXXXXX<char*>(cp); // Part (i)

    p[1] = 'p';                    // F
    p = arr2;                      // G

    return 0;
}
```

(i) The C++-style cast on the line labelled "`Part (i)`" has been replaced by the text "XXXXXXX". What text should replace the "XXXXXXX"? (3)

(ii) Which of the lines labelled A to G contain errors which will prevent this code sample from compiling even when the cast has been included? (3)

(d) Assume that the following C++ source code file is compiled to an executable called "`prog`" and is then executed with the command line "`./prog`" (without the quotation marks).

```
#include <cstdio>
#include <cstring>

int main( int argc, char* argv[] )
{
    char* p = "Filename: ";
    char* q = NULL;

    q = new char[strlen(p)+1+strlen(argv[0])];
    strcpy( q, p );
    strcpy( q+strlen(p), argv[0] );
    printf( "%s", q );
    delete q;

    return 0;
}
```

(i) Assume that the program executes without crashing. What is the output of the program? (4)

(ii) Give details of an error in this code and explain how it could be corrected. (4)

(e) What is the output when the following C++ source code is compiled and executed? (10)

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

class Base
{
protected:
    int i;
public:
    virtual int a() { return i+1; }
    int b() { return i+2; }
    void set( int n ) { i = n; }
};

class Sub : public Base
{
public:
    int a() { return i+3; }
    int b() { return i+4; }
    void set( int n ) { i = n+5; }
};

int main( int argc, char* argv[] )
{
    Sub s;
    Base& r = s;
    Base* p = &s;

    s.set( 10 );
    cout << s.a() << endl;
    cout << s.b() << endl;
    cout << r.a() << endl;
    cout << r.b() << endl;
    r.set( 20 );
    cout << s.a() << endl;
    cout << s.b() << endl;
    cout << r.a() << endl;
    cout << r.b() << endl;
    p->set( 30 );
    cout << r.a() << endl;
    cout << r.b() << endl;
    return 0;
}
```

2 This question is concerned with understanding C code and pointers.

(a) What is the output of the following C source code? (5)

```
#include <stdio.h>
int foo( int x )
{
    static int i = 2;
    int j = 1;
    ++i;
    j+=2;
    return i + j + x;
}

int main( int argc, char* argv[] )
{
    printf( "%d\n", foo( 0 ) );
    printf( "%d\n", foo( 1 ) );
    printf( "%d\n", foo( 2 ) );
    printf( "%d\n", foo( 3 ) );
    printf( "%d\n", foo( 4 ) );
    return 0;
}
```

(b) What is the output of the following C source code? (6)

```
#include <stdio.h>
int main( int argc, char* argv[] )
{
    char* hello = "Hi";
    char* world = "World";
    char* p = hello;
    char* q = world;
    char dest[] = "Destination";
    char* d = dest;

    while( *p || *q )
    {
        if ( *p )
            *d++ = *p++;
        if ( *q )
            *d++ = *q++;
    }
    printf( "%s\n", dest );
    return 0;
}
```

(c) The following source code will compile correctly. What are the twelve numbers which are output when the compiled program is executed? (12)

```
#include <stdio.h>
#include <string.h>

int main( int argc, char* argv[] )
{
    int a[20], b[20];
    int i;

    int* p = a;

    int* q = b+1;

    for ( i = 0 ; i < 19 ; i++ )
    {
        a[i] = i*2;
        b[i] = i*3;
    }
    a[19] = b[19] = 0;

    printf( "%d %d\n", p[0], p[1] );
    printf( "%d %d\n", *q, *q+1 );
    printf( "%d %d\n", *(q+1), *(q+2) );

    while ( *q )
        *q++ = (*p)++;

    printf( "%d %d\n", p-a, q-b );
    printf( "%d %d\n", *p, *q );
    printf( "%d %d\n", b[1], b[2] );

    return 0;
}
```

(d) The following code creates and uses an array of function pointers. What is the output of this code: (7)

```
#include <stdio.h>

int f1( int i ) { return 1; }
int f2( int i ) { return 2; }
int f3( int i ) { return 3; }
int g1( int i ) { return i+1; }
int g2( int i ) { return i+2; }
int g3( int i ) { return i+3; }

int main( int argc, char* argv[] )
{
    int i;
    int (*fnarray[10])(int);
    fnarray[0] = &f3;
    fnarray[1] = &g3;
    fnarray[2] = &f2;
    fnarray[3] = &g2;
    fnarray[4] = &f1;
    fnarray[5] = &g1;
    fnarray[6] = &g2;

    for ( i = 0 ; i < 7 ; i++ )
        printf( "%d\n", (*fnarray[i])(i) );
    return 0;
}
```

3 This question is concerned with the C++ concepts of template functions, operator overloading and casting.

(a) Provide the source code for a function called `product` which will take two integers as parameters and will return an integer result which is the product of the two input values. e.g. `product(3,4)` would return the integer value 12. `product(a,b)`, where `a` and `b` are integer variables, would return `a*b`. (2)

(b) Convert your function from part (a) into a template function which will take two parameters of the same type, to which the * operator can be applied, and will return a value/object of the same type as the parameters.

For example, the following code should then compile (since the * operator is defined for the types `double`, `char` and `int`) and would output the values `1.8` (`double`), `e` (`char`) and `180` (`int`):

```
cout << product( 1.2, 1.5 ) << endl;
cout << product( 'e', (char)1 ) << endl;
cout << product( 12, 15 ) << endl;
```

(8)

(c) Provide implementations for overloads for both the + and += operators which will allow them to be applied to the `Velocity` class, so that the source code below will compile. The + and += operators should be implemented so that the `x` and `y` components are separately added together. The resulting `x` value should be the sum of the `x` components of the two objects and the resulting `y` value should be the sum of the `y` components. (12)

```
class Velocity
{
public:
    double x, y;

    Velocity( double xp, double yp )
    {
        x = xp;
        y = yp;
    }
};

int main( int argc, char* argv[] )
{
    Velocity vel1( 1.0, -2.3 );
    Velocity vel2( 4.5, 5.6 );
    Velocity vel3 = vel1 + vel2;
    // vel3.x == 5.5 and vel3.y == 3.3 at this point
    vel3 += vel1;
    // vel3.x == 6.5 and vel3.y == 1.0 at this point
    return 0;
}
```

(d) What is the difference between the `dynamic_cast` and `static_cast` in C++? Illustrate your answer by giving an example of the use of each and specify what the advantages are of each compared with the other. (8)

4 This question is concerned with C++ concepts, including references and exceptions.

(a) What is the `::` operator used for in C++? Give two examples of its use. (6)

(b) What is the difference between `protected` and `private` member data in C++? (4)

(c) What is the output of the following source code when it is compiled and the resulting program is executed? (10)

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

int a = 1;
int b = 2;

int alter( int x, int& y )
{
    x+=2;
    y+=3;
    return x+y;
}

int main( int argc, char* argv[] )
{
    int& x = a;
    int y = b;

    int i = alter( x, y );
    int j = alter( x, y );

    a+=2;
    b+=3;

    x++;
    y++;

    cout << a << " " << b << endl;
    cout << x << " " << y << endl;
    cout << i << " " << j << endl;

    i = alter( x, y );
    j = alter( x, y );

    a++;
    b++;

    cout << x << " " << y << endl;
    cout << i << " " << j << endl;
    return 0;
}
```

(d) What is the output of the following C++ code? (10)

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

int f1( int i )
{
    if ( (i % 2) == 0 )
        throw i+2;
    return i-1;
}

int f2( int i )
{
    try
    {
        return f1(i) + 3;
    }
    catch( int e )
    {
        return e + 10;
    }
}

int main( int argc, char* argv[] )
{
    for ( int i = 0 ; i < 10 ; i++ )
    {
        try
        {
            switch( i%3 )
            {
            case 0:
                cout << f1( i ) << endl;
                break;
            case 1:
            case 2:
                cout << f2( i ) << endl;
                break;
            }
        }
        catch( int e )
        {
            cout << e << endl;
        }
        catch( ... )
        {
            cout << "?" << endl;
        }
    }
    return 0;
}
```

5 This question is concerned with linked lists, structs, classes, unions.

(a) The following code prints ten numbers when it is executed. If the first three are 1, 2 and 4 respectively, what are the remaining seven numbers? (7)

```
#include <iostream>
#pragma pack(1)
using namespace std;

struct s1
{
    int i;
    short s;
    char c;
};

union u1
{
    int i;
    short s;
    char c;
};

struct s2
{
    union u1 u;
    struct s1 s;
};

union u2
{
    union u1 u;
    struct s1 s;
};

int main( int argc, char* argv[] )
{
    struct s2 s;
    union u2 u;
    cout << sizeof(char) << endl;
    cout << sizeof(short) << endl;
    cout << sizeof(int) << endl;
    cout << sizeof(struct s1) << endl;
    cout << sizeof(union u1) << endl;
    cout << sizeof(union u2) << endl;
    cout << sizeof(s) << endl;
    cout << sizeof(u) << endl;
    cout << sizeof(s.s) << endl;
    cout << sizeof(u.s.s) << endl;
    return 0;
}
```

(b) What is the difference between a `struct` and a `class` in C++? (2)

(c) Consider the following C++ source code for a linked list entry:

```
struct ListEntry
{
    ListEntry* pNext;
    char* pName;
    int iAge;
};
```

What change or changes would be required so that this code would compile on a C compiler? (3)

(d) Assume that the global variable `g_pHead` points to the head of a linked list. Implement the function `AddNew()`, below, that takes two parameters, a string pointed to by `pName` and an integer, `iAge`. `AddNew()` should make a *copy* of the name and age and then store them in a newly allocated `ListEntry`, which is then added to the *head* of the list pointed to by `g_pHead`.

```
ListEntry* g_pHead = NULL;  // Pointer to head of list

void AddNew( const char* pName, int iAge )
{
    // Your code should go here
}
```

(11)

(e) Provide an implementation of a `PrintAll()` function to print the name and age for all of the entries in the list, printing the most recently added entry first. (7)

```
void PrintAll()
{
    // Your code should go here
}
```