The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, AUTUMN SEMESTER 2009-2010

C/C++ for Java Programmers

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 others

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 line of code:

```
char ptr[] = "Hello World"; char a = ptr[1], b = *(ptr+6);
```

For each of the following statements, say whether it is true or false for the above line of code.

(i) The code will compile as ANSI (C89) standard C (e.g. by '`gcc –ansi`')

(ii) An array is created, with the name `ptr`

(iii) An array of 11 elements is created

(iv) The variable a will be initialised to the value '`H`'

(v) The variable b will be initialised to the value '`W`'

(5)

(b) Consider the following source code:

```
struct A
{
    int i;
    int geti() { return i; }
};

int main()
{
    A a;
    return 0;
}
```

Which of the following compilers would compile the code (without raising compilation errors)?

(i) Both a C++ compiler and an ANSI standard C compiler

(ii) A C++ compiler but not an ANSI standard C compiler

(iii) An ANSI standard C compiler but not a C++ compiler

(iv) Neither a C++ nor an ANSI standard C compiler (4)

(c) The following program is supposed to take two command line arguments. The first should specify the filename to open and the second should specify the text to write to the file. The file should be opened, the text written to it and the file closed again. However, the compiler reports one or more errors when it is compiled.

```
#include <stdio.h>
int main( int argc, char* argv[] )
{
    if ( argc < 3 )
    {
        printf( "Usage: %s <filename> <contents>\n", *argv );
        return 1;
    }
    FILE file;
    file.fopen( argv[1], "w" );
    fputs( argv[2], file );
    file.fclose();
    return 0;
}
```

Specify which line or lines of code cause the errors and provide corrected versions of the incorrect line/lines so that the function will work as it should. (7)

(d) The following C source code was successfully compiled using gcc and the resulting program was executed.

```
#include <stdio.h>
#pragma pack(1)

union A
{
    int i;
    char c;
};

struct B
{
    int i;
    char c;
};
```

```
struct C
{
    union A u;
    char c;
};

union D
{
    struct C s;
    union A u;
};

struct E
{
    char x : 3, y : 3;
};

union F
{
    struct E s;
    char c;
};

int main( int argc, char** argv )
{
    printf( "%d %d %d %d %d %d %d %d\n",
        sizeof(char), sizeof(int),
        sizeof(union A), sizeof(struct B),
        sizeof(struct C), sizeof(union D),
        sizeof(struct E), sizeof(union F) );
    return 0;
}
```

Note: the "`#pragma pack(1)`" will, in this case, prevent empty space being included in the `structs/unions`, allowing you to predict the sizes of the types.

This code will output eight numbers, giving the sizes of variables of each type. If the first two numbers which are output are 1 (for `char`) and 4 (for `int`), then what are the remaining six numbers which are output? (6)

(e) The following code sample defines a new function called `strjoin`. `strjoin` should join two strings (`str1` and `str2`) together into a new string (`dest`), the contents of which is the concatenation of the two source strings (i.e. one string followed by the other). For example, joining the strings `"One"` and `"Two"` should result in the string `"OneTwo"`. Although the following sample will compile correctly, it has a bug which means that it does not currently work correctly.

```
char* strjoin( char* str1, char* str2, char* dest )
{
    char* p = dest;
    char* q = str1;

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

    q = str2;
    /* Answer for part (iii) goes here */

    while ( *p++ = *q++ )
        ;
    return dest;
}

int main()
{
    char src1[] = "Hello";
    char src2[] = " World";
    char dest[20];

    strjoin( src1, src2, dest );
    printf( "%s\n", dest );
    return 0;
}
```

(i) What is the problem with the program? (4)

(ii) What is the output of the program at the moment? (2)

(iii) The bug can be corrected by adding an additional (short) line of code at the position which has been identified in the `strjoin` function. What could be added at this point so that the function works as desired? (4)

(f) The following C++ source code defines two structures and a comparison function . The comparison function should determine whether the 'i' data members of two objects of type struct B are identical. It has some errors.

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

struct A { int i, j, k; };
struct B { A a; B* pb; };

bool comparei( struct B* x, const struct B& y )
{
    return x.i == y.i;  // FIRST ERROR – Part (i)
}

int main()
{
    B b1, b2;
    memset( &b1, 0, sizeof(b1) );
    memset( &b2, 0, sizeof(b2) );

    if ( comparei( ?, ? ) ) // SECOND ERROR = Part (ii)
        printf( "b1 and b2 have the same i value.\n" );
    else
        printf( "b1 and b2 have different i values.\n" );
}
```

(i) The comparison within the function `comparei()` (labelled with a comment saying that it is the first error) is supposed to compare the `i` elements of the two objects which are passed in. However, it is incorrect. Write the correct code for this comparison. (i.e. what should appear to the left of the == and what should appear to the right of the == in order to correctly compare the `i` elements of the two objects). (5)

(ii) The function call to compare the two items is also incorrect. What should be passed into the `comparei()` function in order to compare the `i` values of objects b1 and b2. (i.e. what should replace the ?s in the code.) (3)

2 This question is concerned with linked lists and operator overloading. (30 marks)

(a) You are provided with some ANSI standard C (not C++) code to implement a linked list. The structure (in C) for a list element is given below. However, part of the code is missing.

```
typedef struct _LE
{
    char* pFirstName;
    char* pLastName;
    int iAge;
    __A__ pNext;
} ListEntry;
```

What is the missing text which should be inserted into the part labelled __A__? (4)

(b) You are asked to provide an implementation for a C function to allocate the memory required for a new list entry and initialise the data values of the `struct` to copy the strings and integer which are passed in as parameters and store the values in the data members of the `ListEntry`.

```
ListEntry* CreateListEntry(
        char* pFirstName, char* pLastName, int iAge )
{
    __B__ /* The missing lines go here */
    pEntry->iAge = iAge;
    pEntry->pNext = NULL;
    return pEntry;
}
```

Provide the (multiple) missing lines of code which should go into the position labelled __B__ in the function. These should allocate any required memory in a manner that is appropriate for a C program, and initialise the remaining data members of the structure. They should not attempt to link the list item into a list. (6)

(c) A global pointer (g_pHead) has been provided to the head of the list, and a function has been written to link the entry into an existing list, ordering the entries by age. This function uses the existing CreateListEntry() function (from part b) to create the list item and keeps a pointer to the item after which the new item should be added (pPreviousEntry).

```
ListEntry* g_pHead = NULL;

void InsertNewListEntry(
        char* pFirstName, char* pLastName, int iAge )
{
    ListEntry* pNewEntry = CreateListEntry(
                pFirstName, pLastName, iAge );
    ListEntry* pPreviousEntry = g_pHead;

    if ( (g_pHead == NULL)
            || (g_pHead->iAge > iAge) )
    {
        __C__ /* Missing lines for part (i) go here */
        return;
    }

    while( __E__ /* Condition for part (iii) goes here */ )
        pPreviousEntry = pPreviousEntry->pNext;

    __D__ /* Missing lines for part (ii) go here */
}
```

Most of the code has been provided in the implementation above. However some lines have been missed.

(i) What line or lines of code should replace the __C__ in the implementation of the function above in order to link the item into the list correctly? (4)

(ii) What line or lines of code should replace the __D__ in the implementation of the function above in order to link the item into the list correctly? (4)

(iii) The condition for the loop has been excluded. What is the condition which should replace the __E__ in the implementation of the function above, so that the code will compile and execute as expected. (6)

(d) Now assume that the existing program is to be instead compiled as C++ rather than C. (You can assume that any necessary modifications are made to make the files compile as C++, including the inclusion of any necessary header files and any required additional casting of values.)

An overloaded comparison operator (the == operator) is required which will compare whether two list items have the same last name stored in them, so that the following test code reports that both tests have been passed.

```
void CompareTest()
{
    ListEntry* pEntry1 = CreateListEntry("Adam", "Smith", 12 );
    ListEntry* pEntry2 = CreateListEntry("Bill", "Smith", 18 );
    ListEntry* pEntry3 = CreateListEntry("Bill", "Jones", 18 );

    if ( *pEntry1 == *pEntry2 )
        printf( "Test 1 passed\n" );
    else
        printf( "Test 1 failed\n" );

    if ( *pEntry2 == *pEntry3 )
        printf( "Test 2 failed\n" );
    else
        printf( "Test 2 passed\n" );

    ReleaseListEntry( pEntry1 );
    ReleaseListEntry( pEntry2 );
    ReleaseListEntry( pEntry3 );
}
```

Provide an implementation for the operator overloading for the == operator, which will ensure that the correct output is obtained when list items are compared. (6)

3 This question is concerned with understanding operators, understanding C code and exception handling. (30 marks)

(a) In addition to a default (no-parameter) constructor and a destructor, name two other methods which may be created implicitly by a C++ compiler for a C++ `class` if they are needed. (4)

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

```
printf( "%d,%d,%d,%d,%d,%d,%d\n",
    14/3, 5|3, 3<<4, 4&3, 89%3, 4>>1, 0x11 );
```

(7)

(c) What is the output of the following C program? (9)

```
#include <stdio.h>

int i = 1;

int f( int* pi )
{
    int i = *pi;
    return ++i;
}

int g( int& ri )
{
    static int i = ri;
    return i++;
}

int main()
{
    int* pi = &i;
    int& ri = i;

    printf( "%d\n", i++ );
    printf( "%d\n", ri++ );

    i = 10;
```

```
    printf( "%d\n", f(&i) );
    printf( "%d\n", f(&ri) );

    i = 20;
    printf( "%d\n", g(i) );
    printf( "%d\n", g(ri) );

    i = 30;
    printf( "%d\n", *pi++ );
    printf( "%d\n", pi-&i );
    return 0;
}
```

(d) The following code should raise and handle exceptions. Four lines of text are output when it is executed, one line for each iteration of the `for` loop.

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

struct A
{ int a; };

struct B : public A
{ int b; };

int main()
{
    for ( int i = 0 ; i < 4 ; i++ )
    {
        cout << i << " : ";
        try
        {
            switch( i )
            {
            case 0:
                {
                    A a;
                    throw a;
                    break;
```

```
                }
            case 1:
                {
                    B b;
                    throw b;
                    break;
                }
            case 2:
                throw new A;
            case 3:
                throw new B;
            }
        }

        catch( A a )
        {
            cout << "Caught an A" << endl;
        }
        catch( B* pb )
        {
            cout << "Caught a B*" << endl;
        }

        catch( ___XXXXX___ )
        {
            cout << "Caught something else" << endl;
        }
    }
    return 0;
}
```

(i) What is the missing text, which should replace the text ___XXXXX___ in the above code sample, so that the final catch will catch an exception of any type? (2)

(ii) What is the output which will be generated if this code sample is compiled and executed? (8)

4 This question is concerned with function overloading, classes and structs, function pointers, object assignment and construction, and casting in C++. (30 marks)

(a) What is meant by the term 'overloaded function' in C++? (3)

(b) For each of the following code samples, say whether it would compile or not on a standard C++ compiler, and if not then briefly state what the problem is.

Sample 1:

```
class Sample1
{
    Sample1() { i = 4; }
    int i;
};

int main( int argc, char** argv )
{
    Sample1 ob;
    Sample1* p = &ob;
    return 0;
}
```

Sample 2:

```
struct Sample2
{
    Sample2() { i = 4; }
    int i;
};

int main( int argc, char** argv )
{
    Sample2 ob;
    Sample2* p = &ob;
    return 0;
}
```

(5)

(c) What is the output of this C program?

```
int a( int i)
{
    return 2;
}

int b( int i )
{
    return i+1;
}

int main()
{
    int (*f1)(int);
    int (*f2)(int);
    int i, j, k, l, m, n;

    f1 = f2 = &a;
    i = f1(10);

    f1 = &b;
    j = (*f1)(20);

    k = (*f2)( (*f1)( 30 ) );

    l = (*f2)( (*f2)( 40 ) );

    m = (*f1)( (*f2)( 50 ) );

    n = (*f1)( (*f1)( 60 ) );

    printf( "%d,%d,%d,%d,%d,%d\n", i, j, k, l, m, n );
    return 0;
}
```

(6)

(d) What is the output of the following code?

```
#include <iostream>

using namespace std;

class C
{
public:
    C( int i = 1 ) { v = i; }
    C( const C& o ) { v = o.v + 2; }
    C& operator= (const C& o ) {v = o.v + 5; return *this;}
    void out() { cout << v << endl; }
protected:
    int v;
};

int main()
{
    C a, e, f;
    a.out();
    C b(4);
    b.out();
    C c(a);
    c.out();
    C d = b;
    d.out();
    e = b;
    e.out();
    f = b = a;
    f.out();
    return 0;
}
```

(6)

(e) Briefly explain, with short example C++ source code, when and how a `dynamic_cast` would/should be used in C++. Ensure that you explain what advantage(s) it may have over a `static_cast` and in what circumstances. Your answer should illustrate the syntax of how the `dynamic_cast` is used, and what the resulting value from the cast is. (10)

5 This question is concerned with macro definition, template functions, subclassing, and the use of const. (30 marks)

(a) A C or C++ macro is required, to find the (positive) difference between two numbers, so that, for example:

- `DIFF(1,3)` is 2
- `DIFF(3,1)` is 2
- `DIFF(4,-1)` is 5
- `DIFF(-1,-2)` is 1
- `DIFF(-2,-1)` is 1
- `DIFF(1,3) + DIFF(3,1)` is 4

For each of the following macro definitions, state either 'Yes' if it is a correct definition for a macro which would work as described above, or 'No' if it would not work as described.

(i)
```
#define DIFF(x,y) { return (x>y) ? (x-y) : (y-x); };
```

(ii)
```
#define DIFF(x,y) {return((x)>(y)) ? ((x)-(y)) :((y)-(x));}
```

(iii)
```
#define DIFF(x,y) ((x>y) ? (x-y) : (y-x));
```

(iv)
```
#define DIFF(x,y) ((x)>(y)) ? ((x)-(y)) : ((y)-(x))
```

(6)

(b) Write a function which will do the equivalent of the DIFF macro, taking two integer parameters and returning the positive difference between them as an integer. (4)

(c) Convert the function that you provided in part (b) into a template function, so that it will work for any two parameters of a numerical type (where the parameters need not necessarily be the same type, but must be comparable using the > operator) and will return a value of the same type as the first parameter. (6)

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

```
#include <iostream>

using namespace std;

class B
{
protected:
    int i;
public:
    B() : i(10) {setVal(15);}
    int getVal() const { return i; }
    int setVal(int i) { this->i = i; }
};

class S : public B
{
protected:
    int i;
public:
    S() : i(20) {setVal(25);}
    int getVal() const { return i; }
};

int main()
{
    B b;
    S s;
    cout << b.getVal() << endl;
    cout << s.getVal() << endl;

    b.setVal(1);
    cout << b.getVal() << endl;
    s.setVal(2);
    cout << s.getVal() << endl;

    B* pb = &s;
    cout << pb->getVal() << endl;
    S* ps = &s;
    cout << ps->getVal() << endl;
    pb->setVal(3);
    cout << pb->getVal() << endl;
    ps->setVal(4);
    cout << ps->getVal() << endl;
    return 0;
}
```

(8)

(e) One or more of the lines of the following source code file will not compile under a standard C++ compiler. Which of the lines (labelled with letters from A to S) will NOT compile?

You may specify just the letter labels of the erroneous lines and do not need to say why it will not compile.

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

class C
{
public:
    C( int i = 0 ) : v(i) {}

    int get() const     { return v; }

    void set( int i )   { v = i; }

    void out()          { cout << v; }

private:
    int v;
};

int main()
{
    C ob;
    C* pob = &ob;       // Line A
    const C* p1 = &ob;  // Line B
    C* const p2 = &ob;  // Line C
    int i = 1;          // Line D

    i = p1->get();      // Line E
    p1->set( i );       // Line F
    p1->out();          // Line G

    i = p2->get();      // Line H
    p2->set( i );       // Line I
    p2->out();          // Line J
```

```
    i = pob->get();     // Line K
    p2->set( i );       // Line L
    p2->out();          // Line M

    p1 = p2;            // Line N
    p2 = p1;            // Line O

    p1 = pob;           // Line P
    pob = p1;           // Line Q

    p1 = p2;            // Line R
    p2 = p1;            // Line S

    return 0;
}
```

(6)