Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

## Faculty of Engineering, Mathematics and Science

## School of Mathematics

JF Mathematics
JF Theoretical Physics

Trinity Term 2016

### MA1264- Introduction to Programming in C/C++

Thursday, May 5 GOLDHALL 14:00 — 16:00

Prof. C. Ó Dúnlaing

**Instructions to Candidates:**

Attempt 3 questions

**You may not start this examination until you are instructed to do so by the Invigilator.**

1. (a) Convert 2345 and $-5432$ to short integer format (presented as 4 hex digits), and calculate their sum. 'Little endian' is not needed.

   (b) What does (i), (ii), (iii) do, and why?

```
(i)                          (ii)                         (iii)

int i = 0;                   int i = 0;                   int i = 0;

while ( i=0 )                while ( i==0 )               while( i== 0);

{ printf("hello\n");}        { printf("hello\n");}        { printf("hello\n");}
```

   (c) Given

```
double a[4][7];
```

   where `a[0][0]` is at address 1234, (i) How many bytes are occupied by a? (ii) What is `a[2]`? (iii) What is the address of `a[1][3]`?

2. (a) Simulate the following program *carefully.* What does it do, for general $n \geq 0$?

```
#include <stdio.h>
main()
{ int s,i,n;  s = 0; n = 3;
  for (i=0; i<n; ++i)
  { s += 2*i + 1; }
  printf("n is %d, s is %d\n", n, s);
}
```

   (b) Simulate the following program *carefully.* (It has no relation to the previous program.) What does `xx(n)` do, for general $n \geq 0$?

```
#include <stdio.h>
int xx ( int n )
{ if ( n == 0 )
    return n;
  else
    return 2*n+1 - xx ( n-1 );
}
```

```
main()
{ printf ( "xx(4) is %d\n", xx(4)); }
```

(c) i. What does 'little endian' mean?

ii. What does the following program print? What does xxx(x, &y) do, in general?

```
#include <stdio.h>
        // assume ints are 4 bytes, shorts are 2
void xxx( short x, int * y )
{ char * a = ( char *) &x, * b = ( char *) y;
  int i;
  b[0] = a[0]; b[1] = a[1];
  if ( x >= 0 )
    b[2] = b[3] = (char) 0;
  else
    b[2] = b[3] = (char) -1;
}
main()
{ short x = -123; int y;
  xxx ( x, &y );
  printf("x %d, y %d\n", x, y );
}
```

3. (a) Write a C function double ** make_array( int m, int n) which returns an array which can be used as an $m \times n$ array of zeroes.

(b) Write a C++ program which reads a list of integers from standard input, stores their frequency counts in a map, and outputs a frequency table.

The 'map' class template would be very useful.

For example, with input

```
1 2 0 1 0 2 2 0 1 0
```

the output should be

```
0 occurs 4 times
1 occurs 3 times
2 occurs 3 times
```

4. (a) What does the following C program print, and why?

```
#include <stdio.h>
int m = 1, n = 2;
int a ( int n )
{ ++n;
  return n;
}
int b ( int x )
{ ++n;
  return m+n+x;
}
int c ( int x )
{ int m=25;
  return m+n+x;
}
main()
{ int x;
  x = a ( 3 ); printf("x == %d\n", x);
  x = a ( 3 ); printf("x == %d\n", x);
  x = b ( 4 ); printf("x == %d\n", x);
  x = b ( 4 ); printf("x == %d\n", x);
  x = c ( x ); printf("x == %d\n", x);
  x = b ( 4 ); printf("x == %d\n", x);
}
```

(b) Let us define an *index array* of size $n$ to be an array $\{p_0, p_1, \ldots, p_{n-1}\}$, which is a rearrangement of the indexes $0, 1, \ldots, n-1$. Given a class

```
class Index_array
{ public:
    Index_array ( int n );  // construct identity array
    void print ();
    void swap ( int i, int j ); // swap entry[i] with entry[j]
    int  size ();
 private:
    int count;
    int * entry;
};
```

write code for the constructor, print, and swap routines. The constructor should set count to $n$, allocate entry, and initialise entry[i] to $i$ for $0 \leq i \leq n-1$.