UNIVERSITY OF DUBLIN

MA1264-1

TRINITY COLLEGE

Faculty of Science

school of mathematics

JF Maths/TSM **Trinity Term 2015**

Mathematics 1264: C programming and computation

Thursday, May 14 Luce Upper 09.30—11.30

Prof. Colm Ó Dúnlaing

Attempt 3 questions.
Non-programmable calculators may be used. Write the name and model on your answer booklets.

1. (a) Convert 2718 and −668 to 2s complement short integers, and add them as short integers (little endian not required).

   (b) Given the following declarations on a 32-bit-address machine

   ```
   double a[10][10];
   double *b[100];
   ```

   Suppose that `a` begins at address 1000, and `b` begins immediately after `a`.

   i. How many bytes are occupied by the array `a`? By `b`?

   ii. At what address does `b` begin?

   iii. What is the address of `a[2][3]`?

   iv. `a[11][11]` is outside the range of `a`, it actually coincides with `b[i]` for some `i`. Calculate `i`.

ANSWER

```
      169 r 14        10 r9        41 r 12   41 = 2x16+9
16 ) 2718       16 ) 168       16)668         ffff
     16                           64          029c
     111    short: 0a9e            28         fd63
      96                                        1
      158                                      fd64
      144
```

```
0a9e
fd64
0802
```

Array `a`: 800 bytes
`b`: 400 bytes
`b` begins at 1800
`a[2][3]` is at address 1184
`i` is 42

---

2. (a) Write a C function `int count_vowels( char s[])` which counts the number of occurrences of `'a', 'e', 'i', 'o', 'u'` in `s`.

   (b) Write code for the constructor and the two operators (+,*) in the class Complex (obviously meant for complex numbers).

```
typedef class Complex
{
  public:
    Complex ( double, double );
    Complex operator + ( const Complex & other );
    Complex operator * ( const Complex & other );
    void print();
  private:
    double x,y;
} Complex;
```

ANSWER

```
int count_vowels ( char s[] )
{
  int i, count, j;
  char table[] = "aeiou";
  count =0;
  for (i=0; s[i] != '\0'; ++i)
  {
    for (j=0; j<5; ++j)
      if ( s[i] == table[j] )
        ++ count;
  }
  return count;
}
```

```
Complex::Complex ( double a, double b )
{ x = a; y=b;}
Complex Complex::operator + ( const Complex & other )
{  return Complex ( x + other.x, y + other.y );}
Complex Complex::operator * ( const Complex & other )
{ return Complex ( x*other.x-y*other.y, x*other.y+y*other.x ); }
```

---

3. (a) Carefully simulate the following program, showing what it prints.

```
#include<stdio.h>
int xxx( int x, int n )
{ int y;
  if ( n == 0 )
    return x;
  else
  { y = xxx (x,n-1);
    return y*y;
  }
}
main()
{ printf("xxx(3,2) is %d\n", xxx(3,2));
}
```

   (b) What does `xxx(x,n)` compute in general, $n \geq 0$?

   (c) What does the following program print, and why?

```
#include <iostream>
using namespace std;
int n=15;
void b ( int & n )
{  cout << n++ << endl;
}
void c ( int & n )
{  cout << ++n << endl;
}
void c ( double & n )
{  cout << ++n << endl;
}
int main()
{ int n=3; double x=10;
  b(n); c(n); c(x); c(n); c(x);
}
```

ANSWER

```
main
  xxx(3,2)
  | x 3 n 2 y ?
  | xxx(3,1)
  | | x 3 n 1 y ?
  | | xxx(3,0)
  | |   returns 3
  | |
  | resume
  | |         y 3
  | | returns 9
  | |
  resume
  | returns 81
prints
xxx(3,2) is 81
```

The function computes x^{2^n} in general.

The program produces the following output

```
3
5
11
6
12
```

4. (a) Write a full program in C or C++ which reads a single line and prints it in reverse. For example

```
Input:
A quick brown
Output:
nworb kciuq A
```

   (b) Write a full program in C++ which reads text word-by-word from input and prints them in sorted order (one word per line). You may print all the occurrences of words in the text, or print each word just once: either is acceptable.

   Useful STL features:

```
    set<...>::insert(), vector<...>::push_back(), sort (), cin>>...,
    set<...>::iterator
```

ANSWER

Reversing letters:

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

main()
{
  char buffer[200];
  int i,n;

  fgets ( buffer, 200, stdin );
  n = strlen ( buffer ) - 1;
  if ( buffer[n] == '\n' )
    --n;
  for (i=n; i>=0; --i)
    printf("%c", buffer[i]);
  printf("\n");
}
```

Sorted, without repetition:

```
#include <iostream>
#include <set>

using namespace std;

int main()
{
  set <string> strings;
  string word;

  while ( cin >> word )
    strings.insert ( word );

  for ( set<string>::iterator it = strings.begin();
                it != strings.end();  ++it )
    cout << *it << endl;

  return 0;
}
```

Sorted, with repetition:

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
  vector <string> strings;
  string word;

  while ( cin >> word )
    strings.push_back ( word );

  sort ( strings.begin(), strings.end() );

  for ( vector<string>::iterator it = strings.begin();
                it != strings.end();  ++it )
    cout << *it << endl;

  return 0;
}
```
