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 Maths/TP/TSM Trinity Term 2018

Mathematics 1266: C programming

Thursday, May 3 Exam Hall 09:30 — 11:30

Prof. Colm Ó Dúnlaing

Instructions to Candidates:

Attempt 3 questions
Show all work.
Remember to fold down and glue the flap on every answer booklet.

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

1. (a) Convert -3141 to a short integer, giving the answer in hex, little endian.

   (b) Given

   ```
   char hello[] = "hello";
   short *x = (short *) hello;
   ```

   Convert x to decimal. Note: the ascii codes for $a \ldots z$ are $97 \ldots 122$.

   (c) Given

   ```
   int a[10];
   double b[3][3];
   char * c = (char*) a;
   ```

   Assume that a begins at address 1000 and b follows a immediately. The address of $b[1][2]$ coincides with the address of $a[i]$ for some i. Calculate i.

2. (a) Write a ***recursive*** routine void print_binary(int n) which prints n in binary, at 'face value.'. For example, with n==5, the output should be 101. (You may assume that $n > 0$, and it is unnecessary to print a newline.)

   (b) Write an efficient ***recursive*** function

   ```
   double power ( int n, double a );
   ```

   which returns a^n. You may assume $n \geq 0$. Using recursion rather similar to that in print_binary(), the function uses relatively few multiplications.

3. (a) Carefully simulate the following program.

   ```
   #include <stdio.h>
   int xxx ( int m )
   { if ( m <= 10 )
       return m;
     else
     { int x = m%10, y = m/10;
       return x - xxx ( y );
     }
   ```

```
}
main()
{ int m = 123;
  int z = xxx ( m );
  printf("m is %d, m-xxx(m) is %d\n", m, m-z);
}
```

For your information: z is congruent to m mod 11.

(b) Write a complete C program which reads lines from input using `fgets()`, stores copies of these lines in an array `char * string[1000]`, and prints them in reverse order, and separated by blank lines. For example example,

```
           |     should produce
a quick    |    fox
brown      |
fox        |    brown
           |
           |    a quick
```

You can assume that at most 1000 lines will be read.

4. (a) Write a routine `void transpose( double a[2][2], double b[2][2] )` which copies to b the transpose of a. You may assume that a and b are different arrays.

(b) Use it in a careful simulation of the following (which violates the assumption)

```
main()
{ double a[2][2] = {{1,2},{3,4}};
   transpose (a,a);
   printf("%f %f\n%f %f\n", a[0][0], a[0][1], a[1][0], a[1][1]);
}
```

(c) Write a routine `void invert( double a[2][2], double b[2][2] )` which stores the inverse of a in b. You may assume that a is invertible and b is a different array.

Recall

$$\begin{bmatrix} u & v \\ w & x \end{bmatrix}^{-1} = \frac{1}{ux - vw} \begin{bmatrix} x & -v \\ -w & u \end{bmatrix}.$$