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 **Michaelmas Term, 2021**

Mathematics MAU11601: introduction to programming in C *(personalised)*

14 December 2021 **2-5pm**

Prof. Colm Ó Dúnlaing
odunlain@maths.tcd.ie

Instructions that apply to all take-home exams:

1. This is an open-book exam. You are allowed to use your class notes, textbooks and any material which is directly linked from the module's Blackboard page or from the module's webpage, if it has one. You may not use any other resources except where your examiner has specifically indicated in the "Additional instructions" section below. Similarly, you may only use software if its use is specifically permitted in that section. You are not allowed to collaborate, seek help from others, or provide help to others.

2. If you have any questions about the content of this exam, you may seek clarification from the lecturer using the e-mail address provided. You are not allowed to discuss this exam with others. You are not allowed to send exam questions or parts of exam questions to anyone or post them anywhere.

3. Unless otherwise indicated by the lecturer in the "Additional instructions" section, solutions must be submitted through Blackboard in the appropriate section of the module webpage by the deadline listed above. You must submit a single pdf file for each exam separately and sign the following declaration in each case. It is your responsibility to check that your submission has uploaded correctly in the correct section.

Additional instructions for this particular exam:

1. There are 4 questions. Credit will be given to the best 3 answers.
2. Show all work. You get no credit where the details are not shown.

Plagiarism declaration: I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar which are available through https://www.tcd.ie/calendar.

Signature: _____

1. (a) [**5 marks**]. Calculate the short integer representation of 11924 and -1214 (big endian, hexadecimal). Calculate the sum as short integers (big endian).

   (b) [**10 marks**]. Calculate the single precision (32 bit, `float`) encoding of the number $-50.0/601$ giving the result in hexadecimal (little endian).

   (c) [**5 marks**] Given the declarations

   ```
   int a[19]; double b[3][4];
   ```

   suppose `a` starts at address 999 and `b` starts immediately after the end of `a`. (i) Calculate the address of `b[1][2]`. (ii) The address of `a[34]` is actually inside the array `b`. Is its address the same as `b[i][j]` for some i, j?

2. (a) [**10 marks**]. Simulate the program below, using indentation to distinguish the different calls to `xxx()` and the different stack frames, and showing what gets printed.

```
#include <stdio.h>
int xxx ( int m )
{ int par;
  if ( m == 0 )
    return 0;
  else
  { return m%10 - xxx ( m/10 );
  }
}
main()
{ printf("xxx(1234) is %d\n", xxx(1234));
}
```

(b) **[10 marks].** (1) Identify at least 5 errors in the following code

```
#include <stdio.h>
void decr ( char s )
{ for (i=0; s[i] != '\0'; ++i)
    if ( s[i] == '\n' )
      s[i] = '\0';
    else
      ++i;
}
int main ()
{ int n = atoi ( argv[1] );
  char buffer[100];
  for (i=0; i<n; ++i)
  if ( fgets ( buffer, 200, stdin ) == NULL );
  { decr ( buffer );
    printf("%s\n", buffer);
  }
  else
  { printf("end of data\n");
  }
```

3. (a) **[7 marks].** Write a function

```
int square ( int n )
```

(assume $n \geq 0$) which computes and returns the square of `n`, *without using multiplication* but using a for-loop or a while-loop. Hint: $(i+1)^2 - i^2 = i + i + 1$.

(b) **[6 marks].** Give a loop invariant for your `square()` function. You need not show it is correct.

(c) **[7 marks].** Write a *recursive* version of the `square()` function, assuming $n \geq 0$. It should not use a for-loop, nor a while-loop, nor multiplication, relying only on recursion.

4. Write a complete program which reads up to 1000 integers from standard input, storing them in an integer array `int x[1000]`, and calculating and printing (a) the count of odd numbers, the average of the odd numbers, and the variance of the odd numbers, and (b) the count, average, and variance of the even numbers.

   It can be assumed that at least two numbers will be even and at least two odd. For example:

```
With data 6 0 5 -2 -1 -4 17 4 -8 16, the output should be
odd numbers:  count 3 average 7.000000 variance 84.000000
even numbers: count 7 average 1.714286 variance 61.904762
```