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

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

Integrated Computer Science **Trinity Term 2018**
BA (Mod) Computer Science and Business
Year 2 Annual Examinations

Systems Programming I

Wednesday 09th May 2018 **Sports Centre** **09.30-11.30**

Dr. Stephen Farrell

Instructions to Candidates:

Attempt **two** questions. All questions carry equal marks. Each question is scored out of a total of 50 marks. All code should be commented, indented and use good programming style.

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

Exam paper is not to be removed from the venue

Materials Permitted for this examination:

Non-programmable calculators are permitted for this examination – please indicate the make and model of your calculator on each answer book used.

Question 1. Version 4 Internet Protocol (IPv4) addresses can be represented in either "dotted-quad" representation (e.g. "192.0.2.1") or as an unsigned 32-bit integer ("12583425" in decimal, for the previous example).

For a dotted-quad value "A.B.C.D" where A,B,C and D are strings representing decimal integers between zero and 255, the integer representation of the IPv4 address is calculated as:

a*256^3+b*256^2+c*256+d

Where "*" indicates multiplication and "^" indicates exponentation, and "a,b,c,d" are the numbers represented by the strings "A,B,C,D."

In all cases, you must properly handle all errors that can be caused by bad inputs and your functions must not leak any memory. If your code depends on any system/library functions, (which is allowed but not necessary), you must explain what each library function does.

(a) Write a 'C' function that takes a null-terminated string as input, checks if it is a valid IPv4 address in dotted-quad notation and if so returns the unsigned integer representation of the address in the "`addr`" location provided by the caller. If the input string is well formatted, then the function should return zero. In the case of errors the function should return a meaningful non-zero error code.

The prototype for your function must be:

```
int dottedQuad2uint(char *str, unsigned int *addr);
```

[30 marks]

(b) Write a 'C' function that takes an unsigned integer as input and generates the dotted-quad IPv4 address equivalent. The resulting null-terminated string must be returned in a buffer supplied by the caller. On input "`buflen`" contains the size of the buffer; on successful output "`buflen`" must contain the length of the string, including the null-terminator.

The protoype for your function must be:

```
int uint2dottedQuad(unsigned int addr,
                    unsigned char *buf,
                    size_t *buflen);
```

[20 marks]

Question 2. You are supplied with a 'C' file (`isprime.c`) and the corresponding header file (`isprime.h`) that provides a primality checking function for integers. The content of `isprime.h` is as follows:

```
/*!
 * @brief: check if an integer is a prime or not
 * @input: candidate is an integer
 * @return: zero if the candidate is a postive prime number,
 *          non-zero otherwise
 */
int checkprime(int candidate);
```

(a) Write a 'C' or 'C++' program where the `main()` function reads a command line argument, and using the `checkprime()` function determines if the input represents a positive prime number. If it does not, then the program should increment the number until it finds the next prime. The program then presents the results in a simple human-readable format. (Note: unlike in the class assignment, we are only dealing with small prime numbers here so you can directly use the built-in "`int`" type to represent numbers.)

You should include all relevant "`#include`" statements, error handling and provide usage information if incorrect or no arguments are provided.

You may make use of the `atoi()` function which is part of the standard 'C' library by including the `<stdlib.h>` header file in your 'C' program. That is not needed in 'C++' if you "`#include <iostream>`".

[25 marks]

(b) Describe the process you would follow to create, compile, link and test your program. If that process involves creating a `Makefile`, say why, and describe the content of such a `Makefile`. (You don't need to, but can choose to, provide the full content of an actual `Makefile`.)

Describe any limitations of your implementation, e.g. if there are inputs that could lead to unexpected outputs. What useful enhancements might you make to your program if time permitted?

[25 marks]

Question 3. You are given the following program which is deficient in many ways.

```
#include <stdio.h>
struct employee{
    char    name[30];
    int     empId;
    float   salary;
};
// declaration of database insert function
void store2db(struct employee foo);
int main()
{
    struct employee emp; /*declare structure variable*/
    /*read employee details*/
    printf("\nEnter details :\n");
    printf("Name ?:");          gets(emp.name);
    printf("ID ?:");            scanf("%d",&emp.empId);
    printf("Salary ?:");        scanf("%f",&emp.salary);
    /*print employee details*/
    printf("\nEntered detail is:");
    printf("Name: %s"   ,emp.name);
    printf("Id: %d"     ,emp.empId);
    printf("Salary: %f\n",emp.salary);
    /* insert into database*/
    store2db(emp);
}
```

(a) Identify five problems with this program and describe how you would fix those problems.

[15 marks]

(b) Provide your own better program in 'C' or 'C++' that does the same work, but properly. If you don't have time to fully work out code for some parts of your program, then indicate aspects for later improvement via code comments, pseudo-code or in text. (The important part here is to say how to do things properly, not necessarily to get the code 100% correct during the exam.)

[20 marks]

(c) Once completed, how would you test your program, to ensure that it is correct and safe to use?

[15 marks]