Paper Code

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

SAMPLE SAMPLE SAMPLE

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

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

SAMPLE SAMPLE SAMPLE

Systems Programming I

DD MMM YYYY **Venue** **00.00 – 00.00**

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.

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. Domain Names, such as www.tcd.ie are used in the Domain Name System (DNS) and in many other applications. For the purposes of this question, a domain name is valid if it meets the following specification:

- A domain name consists of a sequence of labels separated by the dot character "."
- Each label must be between 1 and 63 characters in length
- The overall name (including dot separators) must be less than 255 characters long
- A domain name may have a trailing dot, e.g. "www.tcd.ie." is valid, as is "www.tcd.ie"
- The special name that has only a dot and no labels represents the root of the DNS naming tree, i.e. "." is a valid DNS name; no other valid DNS name starts with a dot.
- Valid characters for use in labels must be digits from 0 to 9 or the ASCII characters between 'A' and 'Z' (inclusive), in either upper or lower case, or a dash "-" or underscore "_" character
- The "rightmost" (or "last") label (ignoring any trailing dot) must not contain a dash ("-") or underscore ("_") or any digits, unless that label starts with the string "xn--" [For the curious amongst you, this last rule is a simplification of the real one that handles the odd encoding of internationalised top-level domain names.]

In all cases, you must properly handle 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 and returns an integer value that indicates whether or not the string is a valid domain name.

```
int checkDNSValid(char *str);
```

[30 marks]

(b) Write a 'C' function to compare two valid DNS names, represented as null-terminated strings. If either name is invalid then the names can not be considered identical. In the case the names are considered identical the function returns zero. Note that "www.tcd.ie" and "WwW.TcD.IE." are considered identical, even with the trailing dot character.

```
int compareDNSNames(char *str1, char *str2);
```

[20 marks]

Question 2. You are supplied with a 'C' file (`fsize.c`) and the corresponding header file (`fsize.h`) that provides a function to check the size of a file (if the file exists). The content of `fsize.h` is as follows:

```
#include <stdlib.h>
/*!
 * @brief: determine the size of a file
 * @input: fname is the file name
 * @input: fsize is a pointer to a size_t for the
 *         file size result
 * @return: zero for success, non-zero otherwise
 */
int getfsize(char *fname, size_t *fsize);
```

(a) Write a 'C' or 'C++' program where the `main()` function reads a list of command line arguments, each of which ought be a file name, calls the `getfsize()` function and presents the results in a human-readable format. You should include all relevant "`#include`" statements, error handling and provide usage information if incorrect or no arguments are provided. So an example call to the program, if the binary is called "`mainfsize`" and the resulting output, might look like:

```
$ ./mainfsize fsize.h not-there /etc/hosts
Size of fsize.h is 262
Can't stat not-there
Size of /etc/hosts is 2073
```

You can assume `getfsize()` will function correctly when given any string as input, but it is desirable to provide more detailed output for error cases. In the example above, a better program might say that the "`not-there`" file doesn't exist, or that the string provided names a directory and not a file etc.

[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>
#include <stdlib.h> // For exit()
#include <string.h>
int main(int argc,char*argv[])
{
     FILE *fptr;
     char filename[100], c;
     if (argc==2) strcpy(filename,argv[1]);
     else {
          printf("Enter the filename to open \n");
          scanf("%s", filename);
     }
     // Open file
     fptr = fopen(filename, "r");
     if (fptr = NULL) {
          printf("Cannot open file \n");
          exit(0);
     }
     // Read contents from file
     c = fgetc(fptr);
     while (c != EOF) {
          printf ("%c", c);
          c = fgetc(fptr);
     }
}
```

(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]