****

**Faculty of Engineering and Applied Science**

**SOFE 3200U Systems Programming**

**Lab Report 1**

|  |  |
| --- | --- |
| **Group Member 1**  **Name: Jasindan Rasalingam**  **Student ID: 100584595** | **Group Member 2**  **Name: Darron Singh**  **Student ID: 100584624** |
| **Group Member 3**  **Name: Kaushal Patel**  **Student ID: 100586212** | **Group Member 4**  **Name: Ashwin Kamalakannan**  **Student ID: 100584423** |

**Date: September 21, 2017**

**Lab Section CRN: 44209**

# **Questions**

## **ARGS.c**

1. **Explain what argc and \*argv[] represent.**

**argc** – this tells the main function how many arguments are going to be passed to it in the array pointed to by \*argv. By default, argc = 1 because you need at least 1 argument to tell the compiler the file name

**\*argv[]** – is a pointer to an array in memory containing the arguments passed to the main function when the file is run.

1. **For the code snippet below, explain the overview of what the loop does and what variables i and j are keeping track of.**



The code above simply prints the arguments in order as they are given. The ‘I’ variable is used to iterate through each argument. It loops from i=0 to i=argc where argc is the number of arguments given. The ‘j’ variable is used to identify which character we are currently on. So when we index argv[0][0], we are on the first character of the first argument.

The outer for loop is used to iterate through the array of arguments. The inner while loop is used to loop through each individual argument. The condition for the while loop is (argc[i][j] != ‘\0’), this is saying that we should keep looping until we reach the ‘null terminator’. In c language, every string has a null terminator at the end to tell the computer when the string ends. After the while loop exits, we printf(‘\n’), this prints a new line character, which causes the rest of the output to print on the next line.

1. **For the code snippet below, explain the overview of what the loop does and how it is different from the previous loop.**



This code essentially creates the same output as the last piece of code. The difference is that the code block has a second loop inside to print each character of each argument one at a time. While this loop simply prints each argument as a string. Also, the code above adds the \n character on the next line, however this loop adds it in the same line.

**CAPFIX.c**

1. **Explain what each variable listed in the table below represents.**

|  |  |
| --- | --- |
| **Variable** | **Description** |
| \*fpt | Pointer to a file object |
| first[20][30] | Array containing first names, 20 names that are 30 characters long |
| last[20][30] | Array containing last names, 20 names that are 30 characters long |
| i | Indicates the current name in either first[] or last[] |
| j | Indicates the current character of a name in first[] or last[] |
| total | Total is the number of lines read in from the input file |

1. For the code snippet below, explain the overview of what it does.



Total is used to keep track of how many lines are read from the file. The while loop is simply an infinite loop. The if statement starts reading a line from file fpt if there are 2 strings, and puts these values into last[total] and first[total]. The total is incremented each time to keep track of which array index we should write the names into. The fscanf function returns the number of inputs successfully read from the file. If the number of inputs is not 2, then it should break out of the while loop.

1. For the code snippet below, explain what it does in detail.



This selects the current name in the array first[] and selects the first character in that name. It checks if the first character is a valid lowercase letter. If it is, it will remove the lowercase letter and add the uppercase equivalent.

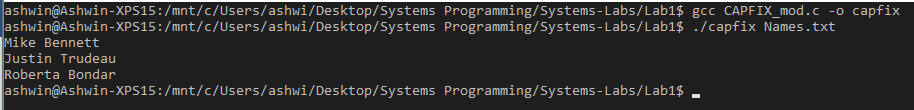
1. For the code snippet below, explain what it does in detail.



This for loop cycles through each character after the first character in the currently selected name in first array. It checks if any of these characters are uppercase. If they are, it removes the uppercase letter and replaces it with the lowercase equivalent.

**Task 3**

We rewrote the CAPFIX.c file to use 2 separate functions. One of them opens the input file provided and reads the data into 2 arrays. The second one takes one of the arrays and fixes the capitalization of each name.



We named the new file ‘CAPFIX\_mod.c’. We compiled it successfully to a file called ‘capfix’. When ran it against Names.txt and it worked perfectly.