**CS2106 Introduction to Operating Systems**

**Lab 1 - Leveling Up on C**

**Answer Book**

Please read the instructions in the main lab sheet before completing this document. Submission deadline is **Sunday 6 February 2022, 11.59 pm**.

|  |  |
| --- | --- |
| **Student 1** | |
| Name: | Wu Xiao Yun |
| Student ID (AxxxxxxY): | A0221772J |
| Group (Bxx): | B06 |
| **Student 2** | |
| Name: | Ang Koon Hwee |
| Student ID (AxxxxxxY): | A0221986W |
| Group (Bxx): | B06 |

Section 1.

**Question 1.1 (1 mark)**

#include <filename> is used for system header files. The preprocessor searches for a file named “filename” in a standard list of system directories (standard C library that is stored in a predefined location).

#include “filename” is used for user-defined header files in local directory. The preprocessor searches for a file named “filename” in the directory that contains the current file, then in the quote directories (directories that have been passed as arguments using the `iquote` option), and then in the same directories used for <filename>.

**Question 1.2 (1 mark)**

The “static” declaration means that the variables remain in memory while the program is running. Static variables are allocated memory in the data segment (not stack).

**Question 1.3 (1 mark)**

`enq` and `deq` are implemented in a different source file (queue.c) from which they are called (lab1p1.c) and there are no function prototypes before the main function in lab1p1.c. Hence the correctness of these function calls cannot be verified.

**Question 1.4 (1 mark)**

`deq` is not declared in the source file from which it is called so the default return type is int. The double value returned from `deq` is being parsed incorrectly, resulting in the incorrect values.

**Question 1.5 (1 mark)**

No warnings. The 2nd for loop now gets the correct values.

Section 2

**Question 2.1 (1 mark)**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Global / Local** | **Address** |
| p1 | G | 0x10222c018 |
| p2 | G | 0x10222c020 |
| p3 | G | 0x10222c028 |
| p4 | G | 0x10222c030 |
| w | L | 0x10222c010 |
| x | L | 0x16dbdb38c |
| y | L | 0x16dbdb388 |
| z | L | 0x16dbdb384 |

**Question 2.2 (1 mark)**

|  |  |
| --- | --- |
| **Variable** | **Location (S, D, T or H)** |
| p1 | D |
| p2 | D |
| p3 | D |
| p4 | D |
| w | D |
| x | S |
| y | S |
| z | S |

How I inferred these answers from Q2.1:

p1-p4 are global variables which are located at the Data segment in the memory. The first 7 digits of the address of w are the same as those of p1-p4, and the address is significantly farther away from the local variables x, y and z. So it is more likely w is located at the Data segment than Stack. Variables x, y and z are located at the Stack segment as they are only available during the function invocation of fun1.

**Question 2.3 (1 mark)**

w is declared as a static variable which is stored at the Data segment of the memory, whose lifetime extends until the end of the program and does not stop when the function call (fun1) has ended. Hence, even though the stack frame for fun1() is discarded when we exit fun1, the value of w is still available (at the Data segment).

**Question 2.4 (1 mark)**

A local variable is only visible within the function that declares it, declaring local variables as static local variables keep their values available in memory throughout the program. On the other hand, global variables are available and visible throughout the source file.

**Question 2.5 (1 mark)**

Change the local variable acc to a static local variable (static int acc = 0;). This ensures that the updated value of acc is available after the function call ends.

Section 3

**Question 3.1 (1 mark)**

malloc() acquires memory space during the execution time (dynamic allocation) so the memory allocated by malloc is stored at the heap segment of the memory. Hence, the address of the memory allocated by malloc is from a completely different range of addresses used by x, y, z (all three are local variables that are stored in the stack segment) and p (global variable stored in the data segment).

**Question 3.2 (1 mark)**

In freeNode function, add free(node->name) before free(node) as memory space was also allocated to the node name. Node name has to be freed to prevent memory leaks.

**Question 3.3 (1 mark)**

gcc -g bintree.c phonebook.c testpb.c -o testpb

**TOTAL: \_\_\_\_\_\_\_\_\_\_\_ / 13**