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 11 February 2024, 1 pm (1300 hrs)**.

Student 1	
Name:	Kenneth Seet
Student ID:	A0258173Y
Group:	B18
Student 2	
Name:	Daniel Wang
Student ID:	A0255689H
Group:	B11

Section 1.

Question 1.1 (1 mark)

The #include <stdio.h> statement in C is used to include the standard input-output library header file. When the angle brackets (<>) are used, the compiler looks for the header file in the system or standard library directories.

Typically, the standard library headers are part of the compiler's installation, and the compiler knows where to find them. The exact location can depend on the compiler and the operating system.

Question 1.2 (1 mark)

The static declaration in the context of a variable declaration means that its scope is limited to the file that it is declared in. This allows it to maintain its value across function calls in the file it is declared in. Functions in the same file can access this variable, but another program's files cannot access them.

Question 1.3 (1 mark)

The error is caused by the enq() and deq() methods lacking a function prototype, so the compiler does not know that the functions are declared. Furthermore, the compiler does not support implicit function declarations, as seen in the error log. The compiler also does not have any knowledge of the memory required to allocate for the parameters for each function call as the type and the number of parameters are unclear, preventing the compiler from making a working executable.

Question 1.4 (1 mark)

void enq(double);

```
double deq();
```

Section 2

Question 2.1 (1 mark)

Variable	Global / Local	Address
p1	G	0x55f290f62018
p2	G	0x55f290f62020
p3	G	0x55f290f62028
p4	G	0x55f290f62030
w	L	0x55f290f62038
x	L	0x7ffc6709996c
y	L	0x7ffc67099968
z	L	0x7ffc67099974

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:

Pointers, which are declared globally outside any function, will be stored in the data segment. Since the address of w, 0x55f290f62038, is near the rest of the addresses in the data segment, from range 0x55f290f62018 to 0x55f290f62030, I can infer that they are in the same segment.

x, y, and z, being local variables, are stored on the stack by default.

Question 2.3 (1 mark)

As observed from question 2.2, w was created in the data segment of the memory, and this allows them to preserve their value between calls to a function as the data segment is reserved for global variables which is separate from the stack which is reserved for local and variables, and the heap, for variables with memory which is dynamically allocated.

The `static int w` declaration ensures that w is a static variable. This means that w will retain its value between calls to fun1 because it is stored in the data segment, and its memory is

not deallocated when fun1 exits. Therefore, even after calling fun2 or exiting the program, the value of w persists.

This allows them to preserve values between calls to a function if the program is running, as they only get deallocated at the end of the program, and not when fun1 or fun2 exits.

Question 2.4 (1 mark)

Declaring a local variable static will allow it to retain its value between calls, but its scope is limited to the function. The initialization of a static local variable occurs only during the first invocation of the function, and subsequent calls use the previously stored value.

Global static variables are visible and accessible throughout the entire file where they are declared. They are initialized once when the program starts and retain their value until the program terminates.

Question 2.5 (1 mark)

We changed line 7 from:
int acc = 0;

to:
static int acc = 0;

This creates a local static variable in the function, allowing it to retain its value from the previous function calls, allowing it to accumulate values.

Section 3

Question 3.1 (1 mark)

The address of the memory allocated by malloc is from a completely different range of addresses used by x, y, z, and p because malloc dynamically allocates memory on the heap, while x, y, z, and p are allocated on the stack. The stack and heap are also two separate regions of memory, and they are not guaranteed to be close to each other.

Question 3.2 (1 mark)

Earlier, the line we were instructed to add into the program was to dynamically allocate memory for the name attribute of the TPerson.

We added a line in freeNode(TPerson *node) to free up node->name:

`free(node->name)`

This was used to free up the dynamically allocated memory for p->name earlier, removing the memory leak.

Question 3.3 (1 mark)

srun gcc -g testlist.c llist.c -o testlist

Section 4

Question 4.1 (1 mark)

This is because the hash is dependent on the filename, and changing the old filename to the new filename would not change its position in the hash table and changing it in place using strcpy(node->filename, new_filename) would prevent the find_file function from finding the file, as its location would be dependent on its old filename, not its updated filename.

Question 4.2 (1 mark)

1. Locate the file in the current hash table using find_file().
2. If the file cannot be found, throw an error, and return early.
3. Else, save the filesize and the startblock attributes from the file in variables.
4. Call delete_file() using the old filename.
5. Call add_file() using the new filename, and the old filesize and startblock attributes which we created variables for beforehand.

TOTAL: _____ / 14