IZMIR INSTITUTE OF TECHNOLOGY

Operating System ASSIGNMENT 2

MOHAMMAD ABU MUSA RABIUL

STUDENT ID: 220201072

Introduction

The assignment objective is to implement **task queue**. Task's operation is either **insert, delete or search** which is performed on a linked list of data. Task queue is also implemented with linked list data structure.

```
/* Struct for list nodes */
struct lst_node_s {
        int data;
        struct lst_node_s* next;
};

/* Struct for task nodes */
struct tsk_node_s {
        int task_num; //starting from 0
        int task_type; // insert:0, delete:1, search:2
        int value;
        struct tsk_node_s* next;
};
```

Figure: data structure of value and task

Program Implementation

The program is implemented in C language. It has three C files (**taskqueue.c, taskqueue.h, and run.c).** In the header file "taskqueue.h", following functions are declared and each function performs different operation.

```
/* List Node functions */

ListNode* createNode(int value); //Create and return a node
with the given data
int isEmpty(ListNode *head); //checks if there is an
element in the list or not
```

```
int insert(int value); //Add given node to the given index
int delete(int value); //Delete first ListNode where given
data occurrs
void printList(ListNode *head); //print the values of each
node in the list
void sortedInsertion(ListNode** head, ListNode* newNode);
// insert a node in sorted order.
int search(int value); // search for a particular value
void deleteAll(ListNode** head); // de-allocating all the
nodes.

/* Task queue functions */

void Task_queue(int n); //generate n random tasks for the
task queue
void Task_enqueue(int task_num, int task_type, int value);
//insert a new task into task queue
int Task_dequeue(); //take a task from task queue and
process that task.
void processTask(TaskNode * task); // process a task
int getTaskSize(TaskNode *head); // get size of the task.
void manageTasks( int n);
```

Inside “taskqueue.c” file, all the functions mentioned above are defined.

Lst_node_s and **tsk_node_s** struct types are renamed with **ListNode** and **TaskNode** respectively. Global pointer variables listHead of type ListNode references the first node of the list and taskHead of type TaskNode references the first task of the task queue in “taskqueue.c” file. The **main** function in “run.c” file calls **manageTasks** function. The **manageTasks** function takes an integer as

argument. Inside the **managerTasks** function, **Task_enqueue** function is called and it creates task queue of size n. Then a loop checks the size of the task queue and inside the loop scope, **Task_dequeue** is called. The objective of **task_queue** function is to take a task from the task queue and process that task by calling **processTask** method. The **processTask** function process the task operation which is either search, insert, or delete operation on data linked list. Upon returning from the **processTask** function, the task is deallocated. At the end, the **manageTask** function deallocated the data linked list to shield against possible memory leak.

During the insert operation, value is checked first inside the data linked list. If value already exist, then it does not insert the value and return false. If value does not exist, then it calls **sortedInsertion** method which objective is to insert data in sorted order (ascending) in the data linked list and return true.

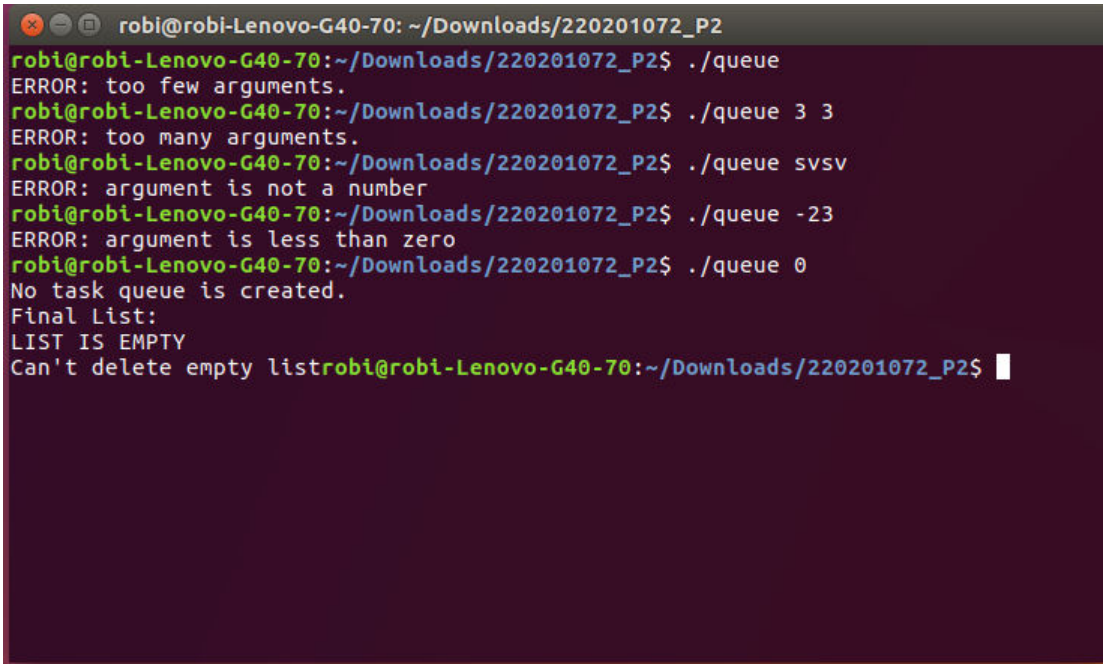
Delete function, delete a node if the data exists otherwise return false. Similarly **search** function, search for a value in the data linked list sequentially and if the value exists return true otherwise return false.

The program also has the following function which does some pre-check for the argument passed into main function.

```
void precheck(char* s, int size);
```

Program Execution

Prechecking



Program execution output: