Name: Bibek Dhungana

Lab 10 - In Lab Assignment

Due end of the lab session

Acknowledge your collaborators or source of solutions, if any. **Submission by the end of the LAB is required.** Please type your answers, handwritten submission will not be accepted. Do all of the following.

1. If X, Y, and Z are inserted into a stack and a queue, what will be the order of removal for the stack? For the queue?

For stack:

We have.

Z
Y
X

Since, stack is last in first out (LIFO) data structure. It is like stack of things or books we see every day. The order of removal is Z, Y and X.

For Queue

We have

X	Y	Z

Since, Queue is First in first out (FIFO). It is like a queue in supermarkets we see every day. The order of removal is X, Y and Z.

2. Name the functions according to their descriptions.

Function	Description
malloc ()	A function that allocates storage space for a single data object that is referenced through a pointer.
calloc()	A function that allocates storage space for an array of objects.
free()	A function that returns the storage space of the heap.

All these functions are defined in stdlib.h header file. malloc() allocate one chunk of memory and return he void pointer to that block.
calloc() also return void pointer to array of object.
free() return that allocated memory from the heap.
We could also use realloc() function to change the size of memory allocation defined by malloc() and calloc() function.

3. Assume the following data type definition and declaration:

```
typedef struct node_s {
int num;
struct node_s *restp;
} node_t;
. . .
node_t *headp, *cur_nodep;
```

Write a for loop header that causes cur_nodep to point in succession to each node of the linked list whose initial pointer is stored in headp. The loop should exit when cur_nodep reaches the end of the list.

It is basically using for loop until the restp pointer points to NULL. This is very crucial that it is required to know when the linked list end.

The prototype can be implemented as follows.

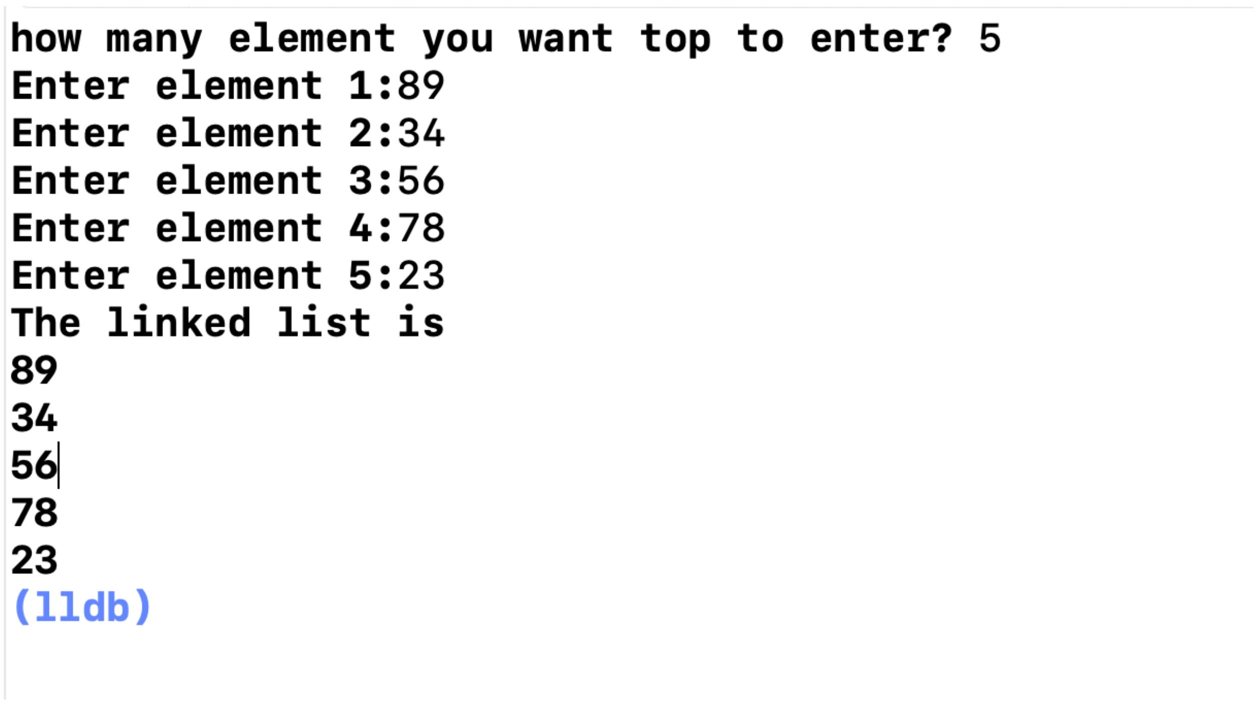
```
for (cur_nodep = headp; cur_nodep != NULL; cur_nodep = cur_nodep->restp){
    printf("%d\n",cur_nodep->num);
}
```

I used this following code to check the result to check the above code and the result was as expected.

Code:

```
#include <stdio.h>
#include <stdlib.h>

//defining the structure
typedef struct node_s{
    int num;
    struct node_s *restp;
}node_t;




int main(int argc, const char *argv[]){
    int num;
    node_t *headp, *cur_nodep, *temp, *last;
    headp = NULL;
    cur_nodep = NULL;
    temp = NULL;

    printf("how many element you want top to enter? ");
    scanf("%d",&num);

    //creating the fisrt node
    headp = (node_t*)malloc(sizeof(node_t));
    printf("Enter element 1:");
    scanf("%d",&headp->num);
    headp->restp = NULL;
    last = headp;


    //adding remaining element
    for(int i = 1; i < num; i++){
        temp = (node_t*)malloc(sizeof(node_t));
        printf("Enter element %d:", i +1);
        scanf("%d",&temp->num);
        last->restp = NULL;
        last->restp = temp;
        last = temp;


    }

    printf("The linked list is\n");
    for (cur_nodep = headp; cur_nodep != NULL; cur_nodep =
cur_nodep->restp){
        printf("%d\n",cur_nodep->num);
    }



    return 0;

}
```

OUTPUT:



```
how many element you want top to enter? 5
Enter element 1:89
Enter element 2:34
Enter element 3:56
Enter element 4:78
Enter element 5:23
The linked list is
89
34
56
78
23
(lldb)
```