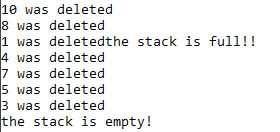
**מעבדה 8 – מת"מ -   
מגישים:  
אורי מלכא – 314862996  
אלן ציפין - 313206062  
  
שאלה 1 קוד  
  
שאלה 2 קוד:  
  
  
  
  
  
שאלה 2 פלט:** **שאלה 3 קוד:**

#include "lab8q1func.h"

void Push(PStack s, int new\_elem) {

//This functions is similar to add to head in a linked list function

if (s->size < NUM) {

Item\* temp; // we create a temp item\* to save pushed node and update the list manager  
 temp = (Item\*)malloc(sizeof(Item));

if (temp == NULL) {

printf("no memo!");

getch();

exit(1);

}

temp->next = s->head;

temp->num = new\_elem;

s->head = temp;

s->size++;

}

else printf("The stack is full!!");

}

int Pop(PStack s, int\* del\_value) {

//this function is similar to delete from head in a linked list.

Item\* temp;

temp = (Item\*)malloc(sizeof(Item));

if (temp == NULL) {

printf("no memo!");

getch();

exit(1);

}

if (s->size > 0)

{

\*del\_value = s->head->num; //we return the popped value via a pointer.

temp = s->head;

s->head = s->head->next;

free(temp); //we free the deleted node

s->size--;

return 1;

}

else {

printf("The stack is empty!! cannot pop!");

return 0;

}

#ifndef \_lab8q1func

#define \_lab8q1func

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define NUM 16

typedef struct Item {

int num;

struct Item\* next;

}Item;

typedef struct Stack {

Item\* head;

int size; //a current number of items

}Stack, \*PStack;

void Push(PStack s, int new\_elem);

/\*

Function name:Push

Input: stack\*, int

Output: void

Function Algorithm:add a new member to list of the stack

\*/

int Pop(PStack s, int\* del\_value);

/\*

Function name:Pop

Input: stack\*, int\*

Output: int

Function Algorithm: delete member from the stack and return the deleted value using int \* del\_value

\*/

#endif

}

#include "lab8q2func.h"

void InitStack(PStack s, int size) {//initialization of a new stack with capacity of size elements

s->top = -1;//Means empty

s->Array = (int\*)malloc(sizeof(int) \* size);

if (s->Array == NULL) {

printf("No memo!!Error!");

getch();

exit(1);

}

s->size = size;

s->count = 0;

}

void Push(PStack s, int new\_elem) { //add a new member to array of the stack

if (s->size != s->count)

{

s->Array[s->top + 1] = new\_elem;

s->top++;

s->count++;

}

else

printf("the stack is full!!");

}

int Pop(PStack s, int\* del\_value) { //delete member from the array of the stack and return the deleted value using int \* del\_value

if (s->top == -1) {

printf("the stack is empty!");

return 0;

}

else {

\*del\_value = s->Array[s->top];

s->top--;

s->count--;

return 1;

}

}

#include "lab8q3func.h"

void Enqueue(PQue Q, int new\_elem) {

Item\* temp;

temp = (Item\*)malloc(sizeof(Item));

if (temp==NULL)

{

printf("no memo");

getch();

exit(1);

}

temp->num = new\_elem;

if (Q->size<NUM) {

if (Q->head) //case if not empty

{

temp->next = NULL;

Q->tail->next = temp;

Q->tail = temp;

}

else //if empty update tail..

{

Q->head = temp;

Q->tail = temp;

}

Q->size++;

}

else

printf("The Que is full.\n");

}

int Dequeue(PQue Q, int\* del\_value) { //delete member from the queue and return the deleted value using int \*del\_value

Item\* temp;

temp = Q->head;

if (Q->size>0) {

\*del\_value = temp->num;

Q->head = Q->head->next;

free(temp); //we free head

Q->size--;

return 1;

}

else {

printf("Queue is empty!");

return 0;

}

}