Imagine you are designing a Game Application where you have to roll a dice which has six faces (a, b, c, d, e and f). Each face has associated value (integer): a=30, b=60, c=40, d=55, e=60 and f=35. Assume you are maintaining two data structures L1 and L2 (Queue or Stack) implemented using linked list. L1 and L2 musts be same data structure. You have to roll the dice 8 times. On each roll, you will get one face on your dice (random value for each roll). Strictly follow the ADT operations.

(a) Add the details of the dice face on List L1 (Acceptance List) only if the most recently inserted value in L1 does not have same (current) face value. If the current face value is the same as the top most value available in L1, then insert the current value into L2 (Rejection List). Count the total acceptance score using L1 and total rejection score using L2 (But don’t traverse the individual element).

(b) Print the final total score earned by playing the game (Total acceptance points) and final total points lost in the game (Total Rejection Points). If Total acceptance points is higher than (Total Rejection Points), than “You are the winner”, otherwise, “You are the looser”.

(c)Empty L1 And L2 using delete operation.

Note: Write functions add(), remove(), TotalfinalScore() by passing proper arguments.

Input:

a b c d e b f f

Output:

Total acceptance points: 280

Total rejection points: 95

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node\* add(struct node \*top, int data){

struct node \*temp=(struct node\*)malloc(sizeof(struct node));

temp->data=data;

temp->next=NULL;

if(top==NULL){

top=temp;

return top;

}

else{

temp->next=top;

top=temp;

return top;

}

}

struct node\* remove\_data(struct node \*top){

if(top==NULL){

printf("There is no node to delete.\n");

}

else if(top->next==NULL){

struct node \*temp=top;

top=NULL;

free(temp);

return top;

}

else{

struct node \*temp=top;

top=top->next;

free(temp);

return top;

}

}

int totalfinalscore(struct node \*top){

int sum=0;

while(top!=NULL){

sum=sum+top->data;

top=remove\_data(top);

}

return sum;

}

void main(){

char arr[8];

int data, i;

struct node \*topL1=NULL, \*topL2=NULL;

printf("Enter the values:\n");

scanf("%s", arr);

for(i=0; i<8; i++){

switch(arr[i]){

case 'a':

data=30;

break;

case 'b':

data=60;

break;

case 'c':

data=40;

break;

case 'd':

data=55;

break;

case 'e':

data=60;

break;

case 'f':

data=35;

break;

}

if(topL1==NULL){

topL1=add(topL1, data);

}

else if(data!=topL1->data){

topL1=add(topL1, data);

}

else{

topL2=add(topL2, data);

}

}

int TotalL1=totalfinalscore(topL1);

int TotalL2=totalfinalscore(topL2);

printf("Total acceptance points: %d", TotalL1);

printf("\nTotal rejection points: %d", TotalL2);

if(TotalL1>TotalL2){

printf("\nYou are Winner\n");

}

else{

printf("\nYou are Looser\n");

}

}