#include <stdio.h>

#include <stdlib.h>

#define size 10

int top = -1;

int \_array[size];

void push(int v){

\_array[++top] = v;

}

void pop(){

top--;

}

int peek(){

return \_array[top];

}

void isEmpty(){

if(top <0) printf("stack is empty");

else printf("stack has %d elements\n",top+1);

}

void print(){

if(top <0) printf("stack is empty");

else {

printf("Elements: \n");

for(int i=0; i<=top; i++){

printf("%d ",\_array[i]);

}

printf("\n");

}

}

int main(){

int ch;

int n,p;

printf("1.push\n2.pop\n3.peek\n4.isEmpty\n5.exit\n");

while(1){

scanf("%d",&ch);

switch(ch){

case 1: scanf("%d",&n);

push(n);

print();

break;

case 2: pop();

print();

break;

case 3: p=peek();

printf("peek function executed %d\n",p);

break;

case 4: isEmpty();

break;

case 5: exit(1);

default: printf("I have no idea what you are trying to do!\n");

exit(1);

};

}

return 0;

}

**OUTPUT**

$ gcc stack\ with\ array.c

$ ./a.out

1.push

2.pop

3.peek

4.isEmpty

5.exit

1 1

1

1 2

1 2

1 3

1 2 3

1 4

1 2 3 4

2

1 2 3

3

Peek function executed 3

# #include <stdio.h>

# #include <stdlib.h>

# #define size 10

# int \_array[size];

# int head=-1,rear=-1;

# void insert(int value){

# if(head == -1 && rear == -1){

# head++; rear++;

# \_array[head] = value;

# }

# else if(rear >= size-1) printf("queue is full\n");

# else \_array[++rear] = value;

# }

# void delete(){

# if(rear == -1) printf("queue is empty\n");

# else rear--;

# }

# int getHead(){

# return head;

# }

# int getRear(){

# return rear;

# }

# void print(){

# for(int i=0; i<=rear; i++){

# printf("%d ", \_array[i]);

# }

# printf("\n");

# }

# int main(){

# int t;

# printf("How many number of inputs?\nMAX SIZE is 10\n");

# scanf("%d",&t);

# if(t<0 || t>10) t = 10;

# while(t--){

# insert(t);

# }

# print();

# insert(size);

# print();}

# OUTPUT

# How many number of inputs? MAX SIZE is 10

# 5

4 3 2 1 0

4 3 2 1 0 10

#include <stdio.h>

#include <stdlib.h>

#include "functions.h"

node head;

int main(){

head = NULL;

int n,t;

printf("Number of inputs?\n");

scanf("%d",&t);

while(t--){

scanf("%d",&n);

insertAtEnd(n);

}

printf("List contains: ");

print();

deleteAtBeginning();

print();

deleteAtEnd();

print();

return 0;

}

**OUTPUT**

$ gcc singlyLinkedList.c

$ ./a.out

Number of Inputs? 5

1 2 3 4 5

List Contains: 1 2 3 4 5

2 3 4 5

2 3 4

#include<stdio.h>

#include<stdlib.h>

struct Node{

int data;

struct Node\* next;

struct Node\* prev;

};

typedef struct Node\* node;

node head;

void insertAtHead(int value){

node temp = (node)malloc(sizeof(node));

temp->data = value;

temp->next = head;

temp->prev = NULL;

head = temp;

}

void insertAtTail(int value){

node temp1 = head;

if(head == NULL){

// printf("List is empty\n");

insertAtHead(value);

return;

}

while(temp1->next != NULL) temp1 = temp1->next;

node temp2 = (node)malloc(sizeof(node));

temp2->next = NULL;

temp2->prev = temp1;

temp2->data = value;

temp1->next = temp2;

}

void deleteAtBeginning(){

node temp = head;

temp = temp->next;

temp->prev = NULL;

free(head);

head = temp;

}

void deleteAtEnd(){

node temp1 = head;

while(temp1->next != NULL){

temp1 = temp1->next;

}

node temp2;

temp2 = temp1->prev;

temp2->next = NULL;

free(temp1);

}

void print(){

node temp = head;

while(temp != NULL){

printf("%d ",temp->data);

temp = temp->next;

}

printf("\n");

}

int getSize(){

node temp = head;

int count=0;

while(temp != NULL){

temp = temp->next;

count++;

}

return count;

}

void printReverse(){

node tempyou = head;

while(tempyou->next != NULL) tempyou = tempyou->next;

node rear = tempyou;

printf("Reverse print function is executing\n");

while(rear != head){

printf("%d ",rear->data);

rear = rear->prev;

}

printf("\n");

}

int main(){

int t,n,c;

printf("Enter the number of inputs\n");

scanf("%d",&t);

while(t--){

scanf("%d",&n);

insertAtTail(n);

}

print();

printf("Do u wanna print this list in reverse order? if yes press 1 \n");

scanf("%d",&c);

if(c==1) printReverse();

printf("Performing delete operation\n");

deleteAtBeginning();

print();

printReverse();

deleteAtEnd();

print();

return 0;

}

**OUTPUT**

$gcc doublyLinkedList.c

$./a.out

Enter the number of inputs 4

1 2 3 4

1 2 3 4

Do u wanna print the list in reverse order if yes press 1

1

4 3 2 1

2 3 4

2 3

#include <stdio.h>

#include <stdlib.h>

#include "functions.h"

node head;

void getTopOfStack(){

node temp = head;

if(temp == NULL) printf("Stack is empty\n");

else{

while(temp->next != NULL){

temp =temp->next;

}

printf("right now TOP of the stack element is %d\n",temp->data);

}

}

void pop(){

node temp1 = head;

if(temp1 == NULL) printf("Stack is empty\n");

else{

for(int i=0; i<getSize()-2; i++){

temp1 =temp1->next;

}

node temp2;

temp2 = temp1->next;

temp1->next = NULL;

free(temp2);

// getTopOfStack();

print();

}

}

void peek(){

node temp = head;

if(temp == NULL) printf("Stack is empty\n");

else{

while(temp->next != NULL){

temp =temp->next;

}

printf("peek function executed\nHere's the value found:%d\n",temp->data);

}

}

void isEmpty(){

if(head == NULL){

printf("Stack is empty");

}

else printf("Seems Like Stack isn't Empty\nThere are %d elements present in stack\n\n\n",getSize());

}

int main(){

head = NULL;

printf("1.push\n2.pop\n3.peek\n4.isEmpty\n5.Get top of the stack\n6.exit\n");

while(1){

int ch,n;

scanf("%d",&ch);

switch(ch){

case 1: scanf("%d",&n);

insertAtEnd(n);

print();

break;

case 2: pop();

print();

break;

case 3: peek();

break;

case 4: isEmpty();

break;

case 5: getTopOfStack();

break;

case 6: exit(1);

break;

default: printf("I have no idea what you are trying to do!\n");

exit(1);

}

}

return 0;

}

**OUTPUT**

$ gcc stack\ with\ LinkedList.c

$ ./a.out

1.push

2.pop

3.peek

4.isEmpty

5.Get top of stack

6.exit.

1 1

1

1 2

1 2

1 3

1 2 3

1 4

1 2 3 4

2

1 2 3

3

Here’s the value found:3

4

There are 3 elements present on the stack

5

Top of the stack is 3

6

$

#include <stdio.h>

#include <stdlib.h>

#include "functions.h"

node head,tail;

void dequeue(){

node temp = head;

if(temp == NULL) printf("Queue is empty\n");

else{

node temp1 = temp->next;

free(temp);

head = temp1;

}

print();

}

int getHead(){ return head->data;}

int getTail(){

node tail = head;

while(tail->next != NULL){

tail = tail->next;

}

return tail->data;

}

int main(){

head = tail = NULL;

int ch,n;

printf("1.enqueue 2.dequeue 3.Get head and rear 4.exit\n");

while(1){

scanf("%d",&ch);

switch(ch){

case 1: scanf("%d",&n);

insertAtEnd(n);

print();

break;

case 2: deleteAtBeginning();

print();

break;

case 3: printf("Head points to %d\n",getHead());

printf("Tail points to %d\n",getTail());

break;

case 4: exit(1); break;

default: printf("I have no idea what are you trying to do!\n");

}

}

return 0;

}

**OUTPUT**

1.enqueue 2.dequeue 3.Get head and rear 4.exit

1 1 1 2 1 3 1 4

1 2 3 4

2

2 3 4

3

Head points to 2 rear points to 4

4

$

#include<stdio.h>

#include<stdlib.h>

struct Node{

int data;

struct Node\* next;

struct Node\* prev;

};

typedef struct Node\* node;

node head,tail;

void insertAtHead(int value){

node temp = (node)malloc(sizeof(node));

temp->data = value;

temp->next = head;

temp->prev = NULL;

head = temp;

}

void insertAtTail(int value){

node temp1 = head;

if(head == NULL){

// printf("List is empty\n");

insertAtHead(value);

return;

}

while(temp1->next != NULL) temp1 = temp1->next;

node temp2 = (node)malloc(sizeof(node));

temp2->next = NULL;

temp2->prev = temp1;

temp2->data = value;

temp1->next = temp2;

}

void deleteAtBeginning(){

node temp = head;

temp = temp->next;

temp->prev = NULL;

free(head);

head = temp;

}

void deleteAtEnd(){

node temp1 = head;

while(temp1->next != NULL){

temp1 = temp1->next;

}

node temp2;

temp2 = temp1->prev;

temp2->next = NULL;

free(temp1);

}

void print(){

node temp = head;

while(temp != NULL){

printf("%d ",temp->data);

temp = temp->next;

}

printf("\n");

}

int getSize(){

node temp = head;

int count=0;

while(temp != NULL){

temp = temp->next;

count++;

}

return count;

}

int main(){

head = NULL;

tail = NULL;

int t,n,m;

scanf("%d",&t);

while(t--){

scanf("%d",&n);

if(n%2==0) insertAtHead(n);

else insertAtTail(n);

}

print();

deleteAtEnd();

print();

deleteAtBeginning();

print();

return 0;

}

**OUTPUT**

$ gcc dequeue\ with\ LinkedList.c

$ ./a.out

4

1 2 3 4

4 2 1 3

4 2 1

2 1

$

struct Node {

int data;

struct Node\* next;

};

typedef struct Node\* node;

node head;

node createNewNode(){

return (node)malloc(sizeof(node));

}

void insertAtBeginning(int x){

node current = createNewNode();

current->data = x;

current->next = head;

head = current;

}

void insertAtEnd(int x){

if(head == NULL){

insertAtBeginning(x);

}

else{

node temp1 = head;

while(temp1->next != NULL){

temp1 = temp1->next;

}

node temp2 = createNewNode();

temp2->data = x;

temp2->next = NULL;

temp1->next = temp2;

}

}

void deleteAtBeginning(){

node temp = head;

temp = temp->next;

free(head);

head = temp;

}

int getSize(){

node temp = head;

int count=0;

while(temp != NULL){

temp = temp->next;

count++;

}

return count;

}

void deleteAtEnd(){

node temp1 = head;

while(temp1->next != NULL)temp1 = temp1->next;

node temp2 = temp1->next;

temp1->next = NULL;

free(temp2);

}

void print(){

node temp = head;

while(temp != NULL){

printf("%d ",temp->data);

temp = temp->next;

}

printf("\n");

}