***// CIRCULAR LINKED LIST***

#include <stdio.h>

#include <stdlib.h>

*// Node of the circular linked list.*

typedef struct Node

{

int data;

struct Node \*next;

} Node;

*// Handle of the list.*

*// Head points to the first node in the list.*

*// Tail points to the last node in the list.*

typedef struct List

{

Node \*head;

Node \*tail;

int length;

} List;

*// Initializes a circular linked list.*

List\* initialize\_list();

*// Creates a node and stores the data in it.*

Node\* create\_node(int data);

*// Inserts data at the head of the list.*

void insert\_head(List\* cll, int data);

*// Deletes the node at the head position. No operation if list is empty.*

void delete\_head(List\* cll);

*// Swaps the first(Head) and last(Tail) element.*

void swap\_first\_and\_last(List\* cll);

*// Prints the data present in the safe node according to the josephus problem.*

int josephus(List\* cll, int k);

*// Prints the entire list. Prints "EMPTY" if the list is empty.*

*void display(List\* cll);*

*// Deallocates resources held by the list.*

void destroy\_list(List\* cll);

int main()

{

List\* cll = initialize\_list();

int ele, choice, pos, k;

do

{

scanf("%d",&choice);

switch(choice)

{

// Insert at Head.

case 1:

scanf("%d",&ele);

insert\_head(cll,ele);

break;

// Delete at Head.

case 2:

delete\_head(cll);

break;

// Josephus problem.

case 3:

scanf("%d",&k);

ele = josephus(cll,k);

printf("%d\n",ele);

break;

// Swap first and last element.

case 4:

swap\_first\_and\_last(cll);

break;

// Print entire list.

case 5:

display(cll);

break;

}

}

while (choice != 0);

destroy\_list(cll);

return 0;

}

List\* initialize\_list()

{

List \*list=(List\*)malloc(sizeof(List));

list->head=NULL;

list->tail=NULL;

list->length=0;

return list;

}

Node\* create\_node(int data)

{

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

temp->data=data;

temp->next=NULL;

return temp;

}

void insert\_head(List\* cll, int data)

{

Node \*temp=create\_node(data);

if(cll->head==NULL)

{

cll->head=temp;

cll->tail=temp;

temp->next=cll->head;

}

else

{

cll->tail->next=temp;

temp->next=cll->head;

cll->head=temp;

}

++(cll->length);

}

void delete\_head(List\* cll)

{

Node \*temp=cll->head;

if(temp==NULL || (cll->length<=0))

return -1;

else if(temp->next==cll->head)

{

free(temp);

temp=NULL;

cll->head=NULL;

cll->tail=NULL;

}

else

{

cll->head=temp->next;

cll->tail->next=temp->next;

free(temp);

temp=NULL;

}

--(cll->length);

}

void swap\_first\_and\_last(List\* cll)

{

Node \*temp=cll->head;

int x=0;

if((temp!=NULL) && (cll->head==cll->tail->next) && cll->head->data!=cll->tail->data)

{

Node \*q=cll->tail;

x=temp->data;

temp->data=q->data;

q->data=x;

}

}

int josephus(List\* cll, int k)

{

Node \*temp=cll->head;

Node \*p=NULL;

for(int i=0;i<=k;i++)

{

p=temp;

temp=temp->next;

}

while(cll->length>1)

{

p->next=temp->next;

free(temp);

temp=NULL;

p=p->next;

temp=p->next;

--cll->length;

}

cll->head=p;

cll->tail=p;

return p->data;

}

void display(List\* cll)

{

Node \*temp=cll->head;

if(cll->head==NULL)

printf("EMPTY");

else

{

while(temp->next!=cll->head)

{

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

temp=temp->next;

}

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

}

printf("\n");

}

void destroy\_list(List\* cll)

{

Node \*temp=cll->head;

Node \*q=cll->head->next;

if(cll->head==NULL)

free(cll);

else if(cll->length==1)

{

free(temp);

free(cll);

}

else

{

while(temp!=cll->head)

{

free(q);

q=temp;

temp=temp->next;

}

}

}