Exercise 7-Doubly and Circular linked List

Name: A H K Swarun
21BPS1252

Ques 1:

Implement a Doubly linked and perform Insert, delete and display operations. (Insertion and deletion at beginning, middle and end)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node *llink;
struct node *rlink;
};
struct node *head = NULL;
struct node *newnode;
struct node *temp;
void create(int x)
{
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = x;
newnode->llink = NULL;
newnode->rlink = NULL;
if (head == NULL)
{
head = newnode;
temp = newnode;
}
else
{
temp->rlink = newnode;
newnode->llink = temp;
temp = temp->rlink;
}
}
void insertatbeg(int x)
{
```

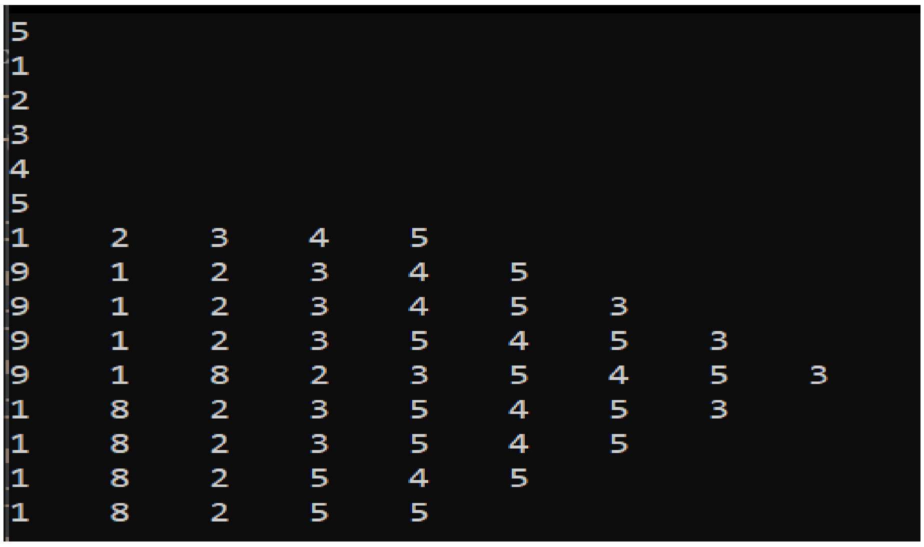
```
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = x;
newnode->llink = NULL;
head->llink = newnode;
newnode->rlink = head;
head = newnode;
}
void insertatend(int x)
{
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = x;
newnode->llink = NULL;
newnode->rlink = NULL;
temp = head;
while (temp->rlink != NULL)
{
temp = temp->rlink;
}
temp->rlink = newnode;
newnode->llink = temp;
newnode->rlink = NULL;
}
void insertatmid(int x)
{
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = x;
newnode->llink = NULL;
newnode->rlink = NULL;
struct node *fast = head;
struct node *slow = head;
struct node *fwd;
while(fast != NULL && fast->rlink != NULL)
{
fast = fast->rlink->rlink;
slow = slow->rlink;
}
fwd = slow->rlink;
newnode->rlink = fwd;
newnode->llink = slow;
slow->rlink = newnode;
fwd->llink = newnode;
}
void insertatpos(int x,int loc)
{
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = x;
```

```
newnode->llink = NULL;
newnode->rlink = NULL;
temp = head;
for (int i = 1; i<loc; i++)
{
temp = temp->rlink;
}
newnode->rlink = temp->rlink;
temp->rlink->llink = newnode;
temp->rlink = newnode;
newnode->llink = temp;
}
void deleteatbeg()
{
struct node *denode;
denode = head;
head = head->rlink;
free(denode);
}
void deleteatend()
{
temp = head;
while (temp->rlink != NULL)
{
temp = temp->rlink;
}
struct node *denode;
denode = temp;
temp = temp->llink;
temp->rlink = NULL;
denode->llink = NULL;
free(denode);
}
void deleteatmid()
{
struct node *fast = head;
struct node *slow = head;
while (fast != NULL && fast->rlink != NULL)
{
fast = fast->rlink->rlink;
slow = slow->rlink;
}
struct node *denode;
denode = slow;
slow->llink->rlink = slow->rlink;
slow->rlink->llink = slow->llink;
```

```
free(denode);
}
void deleteatpos(int x)
{
struct node *denode;
temp = head;
while (temp->data != x)
{
temp = temp->rlink;
}
denode = temp;
temp->llink->rlink = temp->rlink;
temp->rlink->llink = temp->llink;
free(denode);
}
void display()
{
temp = head;
while (temp != NULL)
{
printf("%d    ", temp->data);
temp = temp->rlink;
}
printf("\n");
}
int main()
{
int n;
scanf("%d", &n);
for (int i =0; i<n; i++)
{
int x;
scanf("%d", &x);
create(x);
}
display();
insertatbeg(9);
display();
insertatend(3);
display();
insertatmid(5);
display();
insertatpos(8,2);
display();
deleteatbeg();
display();
```

```
deleteatend();
display();
deleteatmid();
display();
deleteatpos(4);
display();
}
```

OUTPUT:

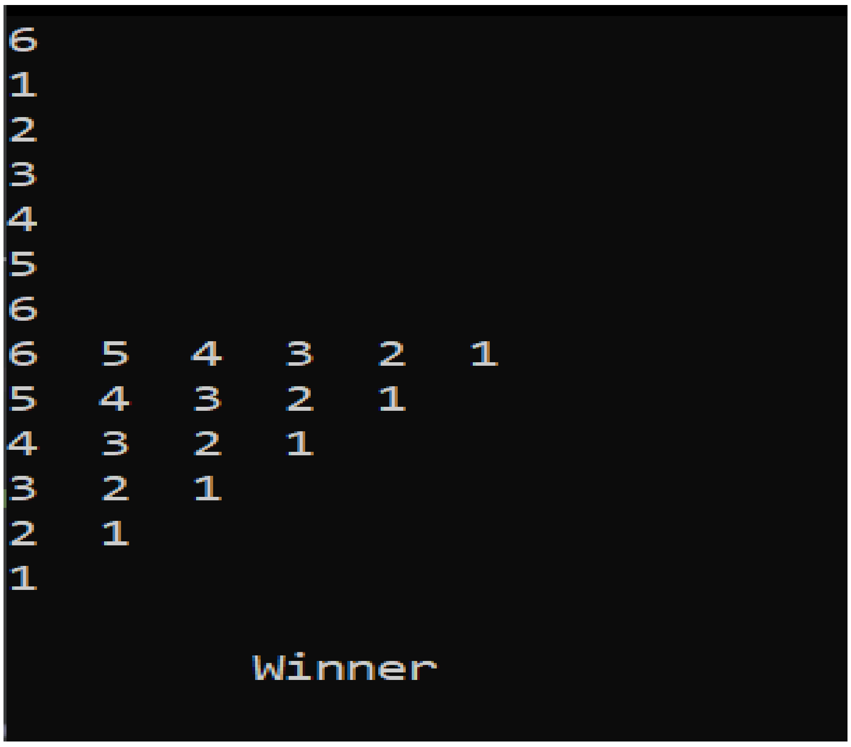


```
5
1
2
3
4
5
1    2    3    4    5
9    1    2    3    4    5
9    1    2    3    4    5    3
9    1    2    3    5    4    5    3
9    1    8    2    3    5    4    5    3
1    8    2    3    5    4    5    3
1    8    2    3    5    4    5
1    8    2    5    4    5
1    8    2    5    5
```

Ques 2:

Consider the game of "passing the doll" where N players sit in a ring and a "doll" is being passed around. The doll continues to do the rounds until the music stops. At the point of time when the music stops, the person holding onto the doll is eliminated from the ring. After elimination, the doll again begins its rounds from the next person. The game ends when there is only one person left who is the winner. First, identify an appropriate data structure to implement this game, then write an algorithm and determine the winner. (Implement using Circular linked list)

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
struct node *prev;
struct node *next;
int data;
};
struct node *head;
void insertion_beginning();
void deletion_beginning();
void display();
int main ()
{
int n;
scanf("%d", &n);
for (int i=0; i<n; i++)
{
int x;
scanf("%d", &x);
insertion_beginning(x);
}
display();
for (int i = 0; i<n; i++)
{
deletion_beginning();
display();
}
}
void insertion_beginning(int item)
{
struct node *ptr,*temp;
ptr = (struct node *)malloc(sizeof(struct node));
ptr->data=item;
if(head==NULL)
{
head = ptr;
ptr -> next = head;
ptr -> prev = head;
}
else
{
temp = head;
while(temp -> next != head)
{
temp = temp -> next;
}
temp -> next = ptr;
ptr -> prev = temp;
head -> prev = ptr;
ptr -> next = head;
head = ptr;
}
}
void deletion_beginning()
{
struct node *temp;
if(head == NULL)
{
printf("\n UNDERFLOW");
}
else if(head->next == head)
{
head = NULL;
free(head);
}
else
{
temp = head;
while(temp -> next != head)
{
temp = temp -> next;
}
temp -> next = head -> next;
head -> next -> prev = temp;
free(head);
head = temp -> next;
}
}
void display()
{
struct node *ptr;
ptr=head;
if(head == NULL)
{
printf("\n\tWinner\t\n");
}
else
{
while(ptr -> next != head)
{
printf("%d  ", ptr -> data);
ptr = ptr -> next;
}
printf("%d  ", ptr -> data);
}
printf("\n");
}
```



```
6
1
2
3
4
5
6
6   5   4   3   2   1
5   4   3   2   1
4   3   2   1
3   2   1
2   1
1

          Winner
```