Data Structures and Algorithms Essential Program

DAY 6 | ASSIGNMENT **EMAIL:pavanibhavya77@gmail.com**

1.)Write a program implementing insert, delete and display operation of Circular Queue.
Sol.)

INSERTION

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node *next;
}*head;
void createlist(int n);
void displaylist();
void insertAtBegining(int data);
void insertAtPos(int data,int position);
int main(){
int n,data,choice=1;
head=NULL;
while(choice!=0){
printf("**CLL**");
printf("1.create list\n2.display\n3.insert at beginning\n4.insert at position\n4.exit\n");
printf("enter choice");
scanf("%d",&choice);
switch(choice){
case 1:printf("enter total number of nodes");
scanf("%d",&n);
createlist(n);
break;
case 2:displaylist();
break;
case 3:printf("enter data to be inserted at beginning");
scanf("%d",&data);
insertAtBegining(data);
break;
case 4:printf("enter node position:");
scanf("%d",&n);
printf("enter data");
insertAtPos(data,n);
break;
case 4:break;
default:printf("invalid choice");
}
```

```
printf(“\n”);
}
}
void createlist(int n){
int i,data;
struct node *prevNode,*newNode;
if(n>=1){
head=(struct node *)malloc(sizeof(struct node));
printf(“enter data”);
scanf(“%d”,&data);
head->data=data;
head->next=NULL;
prevNode=head;
for(i=2;i<=n;i++){
newNode=(struct node*)malloc(sizeof(struct node));
printf(“enter data %d”,i);
scanf(“%d”,&data);
newNode->data=data;
newNode->next=NULL;
prevNode=newNode;
}
prevNode->next=head;
printf(“CLLcreated”);
}
}
void displaylist(){
struct node *current;
int n=1;
if(head=NULL){
printf(list is empty\n”);
}
else{
current =head;
printf(“data in the list:\n”);
do{
printf(“data %d=%d\n”,n,current->data);

current=current->next;
n++;
}
while(current!=head);
}
}
void insertAtBegining(int data){
```

```
struct node *newNode,*current;

if(head==NULL){
printf(“list is empty”);
}
else{
new node=(struct node *)malloc(sizeof(struct node));
newNode->data=data;
newNode->data=data;
newNode->next=head;
current=head;
while(current->next!=head){
current=current->next;
}
current->next=newNode;
head=newNode;
printf(“node inserted”);
}
}
void insertAtPos(int data,int pos){
struct node *newNode,*current;
int i;
if(head==NULL){
printf(“list empty”);
}
else if(pos==1){
insertAtbegining(data);
}
else{
newNode=(struct node *)malloc(sizeof(struct node));
newNode->data=data;
current=head;
for(i=2;i<=pos-1;i++){
current =current->next;
}
newNode->=current->next;
current->next=newNode;
printf(“node inserted”);
}
}
```

DELETION

```
#include<stdio.h>
#include<stdlib.h>
struct node{
```

```
int data;
struct node *next;
}*head;
void delete(struct node **head,int key);
int main(){
int n,key,data,choice;
struct node *head=NULL;
}
void delete(struct node **head,int key){
printf(“enetr key”);
scanf(
int i,count;
struct node*prev,*cir;
if(*heaad==NULL){
printf(“list empty”);
}
count =0;
cur =*head;
prev=cur;
while(prev->next!=*head)
{
prev=prev->next;
Count++;
}
i=0;
while(i<=count){
if(cur->data==key){
if(cur->next!=cur)
prev->next=cur->next;
else
prev->next=NULL;
if(cur==*head)
*head=prev->next;
free(cur);
if(prev!=NULL)
cur=prev->next;
else
cur=NULL;
}
else{
prev=cur;
cur->next;
}
i++;
```

}}

3.)Implement push, pop and find the minimum element in a stack in O(1) time complexity.

```
#include <bits/stdc++.h>
using namespace std;

struct MyStack
{
    stack<int> s;
    int minEle;

    void getMin()
    {
        if (s.empty())
            cout << "Stack is empty\n";

        else
            cout <<"Minimum Element in the stack is: "
                 << minEle << "\n";
    }

    void peek()
    {
        if (s.empty())
        {
            cout << "Stack is empty ";
            return;
        }

        int t = s.top(); // Top element.

        cout << "Top Most Element is: ";

        // If t < minEle means minEle stores
        // value of t.
        (t < minEle)? cout << minEle: cout << t;
    }

    void pop()
    {
        if (s.empty())
        {
            cout << "Stack is empty\n";
            return;
```

```
        }

        cout << "Top Most Element Removed: ";
        int t = s.top();
        s.pop();

        if (t < minEle)
        {
            cout << minEle << "\n";
            minEle = 2*minEle - t;
        }

        else
            cout << t << "\n";
    }


    void push(int x)
    {

        if (s.empty())
        {
            minEle = x;
            s.push(x);
            cout <<  "Number Inserted: " << x << "\n";
            return;
        }

        // If new number is less than minEle
        if (x < minEle)
        {
            s.push(2*x - minEle);
            minEle = x;
        }

        else
           s.push(x);

        cout <<  "Number Inserted: " << x << "\n";
    }
};

int main()
{
```

```
    MyStack s;
    s.push(3);
    s.push(5);
    s.getMin();
    s.push(2);
    s.push(1);
    s.getMin();
    s.pop();
    s.getMin();
    s.pop();
    s.peek();

    return 0;
}
```