week-4

15/01/24 // Circular queue

```
#include <stdio.h>
#include<stdlib.h>
#define size 50
int queue[size];
int rear = -1;
int front = -1;
int isFull()
{
  if (front == (rear+1)%size)
  {
    return 0;
  }
  else
  {
    return -1;
  }
}
int isEmpty()
{
  if (front == -1 && rear == -1)
  {
    return 0;
  }
  else
  {
    return -1;
  }
}
Void Enqueue(int x)
{
  int item;
  if (isFull() == 0)
  {
    printf("Queue overflow\n");
    return;
```

```
else
{
  if (IsEmpty() == -0)
  {
    front=0;
    rear=0;
  }
  else
  {
    rear=(rear+1)%size;
  }
  Q[rear]=x;
}
}
int Dequeue()
{
  int x;
  if (IsEmpty() == 0)
  {
    printf("Queue underflow\n");
  }
else
{
  if (front= rear)
  {
    x = queue[front];
    front=-1;
    rear=-1;
  }
else
{
  x = queue[front];
  front = (front+1)%size;
}
return x;
```

```
void display()
{
int i;
if (isEmpty() == 0)
{
printf("Queue is empty\n");
}
else
{
printf("Queue elements:\n");
for (i = front; i != rear; i = (i+1) % size)
{
printf("%d\n", queue[i]);
}
printf("%d\n", queue[i]);
}
void main()
{
int choice, x, b;
while(1);
{
printf("1.Enqueue, 2.Dequeue 3.display 4.exit\n");
printf("Enter your choice");
scanf("%d", &choice);
switch(choice)
{
case 1: printf("Enter the number to be inserted\n");
scanf("%d", &x);
enqueue(x);
break;
case 2: b = dequeue();
printf("%d was removed from the
queue\n", b);
break;
```

```
case 3: Display();
        break;
case 4: exit(1);
default: printf("Invalid choice");
}
}
```

Output

1 Enqueue 2 dequeue 3. display 4. exit
Enter your choice : 1
Enter the number to be inserted
23
1. Enqueue 2. Dequeue 3 display 4. exit
Enter the number to be inserted
45
1. Enqueue 2. Dequeue 3. display 4. exit
Enter the number to be inserted
28 1. Enqueue 2. dequeue 3 display 4. exit
Enter the choice : 2
Enter the no. of to be deted
45
Enter your choice 3. Queue elements
28 28
Enter your choice 4

Week-4

11/9/24 // Circular linked list

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
  int data;
  struct Node * next;
};
struct Node * createNode(int value)
{
  struct Node * newNode = (struct Node*) malloc
  (sizeof(struct Node));
  if(newNode == NULL)
  {
    printf("memory allocation failed\n");
    exit(1);
  }
  newNode -> data = value;
  newNode -> next = NULL;
  return newNode;
}
struct Node * insertAtBeginning(struct Node * head,
int value)
{
  struct Node * newNode = createNode(value);
  newNode -> next = head;
  return newNode;
}
struct Node * insertAtAnyPos(struct Node * head,
                             int value, int pos)
{
  struct Node * newNode = createNode(value);
  struct Node * temp = head;
  if(pos == -1) {
    newNode -> next = head;
  return newNode;
```

```
}
int i;
for (i=1; i<pos-1; i++)
{
  temp = temp->next;
}
newNode->next = temp->next;
temp->next = newNode;
return head;
}
struct Node* insertAtEnd(struct Node* head,
int value)
{
  struct Node* newNode = createNode(value);
  if(head==NULL)
  {
    return newNode;
  }
struct Node* temp = head;
while(temp->next!=NULL
{
  temp = temp->next;
}
temp->next = newNode;
return head;
}
void displayList(struct Node* head)
{
  struct Node* temp = head;
  while(temp!=NULL)
  {
    printf("%d->", temp->data);
    temp = temp->next;
  }
```

Segmentation fault

```
int main()
{
struct node *head = NULL;
int choice, value, pos;
while(1)
{
printf("1. Insert at end\n");
printf("2. Insert at beginning\n");
printf("3. Insert at any position\n");
printf("4. Display list\n");
printf("5. Exit\n");
printf("Enter your choice:");
scanf("%d", &choice);
switch(choice)
{
case 1: printf("Enter a new value to be inserted");
scanf("%d", &value);
head = insertAtEnd(head, value);
break;
case 2: printf("Enter the value to be inserted");
scanf("%d", &value);
head = insertAtBeginning(head, value);
break;
case 3: printf("Enter the value to be inserted");
scanf("%d", &value);
printf("Enter the position");
scanf("%d", &pos);
head = insertAtAnyPos(head, value, pos);
break;
case 4: displayList(head);
break;
case 5: exit(0);
default: printf("Invalid choice");
}
}
return 0;
}
```

Output

1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Exit

Enter your choice: 2
Enter the value to be inserted: 12

Enter your choice: 2
Enter the value to be inserted: 13

Enter your choice: 4
13 → 12 → NULL

Enter your choice: 1
Enter the value to be inserted: 23

Enter your choice: 4
13 → 12 → 23 → NULL

Enter your choice: 3
Enter the value to be inserted: 12
Enter the position to insert

Enter your choice: 4
13 → 12 → 12 → 23 → NULL