Circular Queue using array

#include <stdio.h>

#include <stdlib.h>

int n;

int front = -1;

int rear = -1;

void enqueue(int \*q)

{

  int x;

  if ((rear + 1) % n == front)

  {

    printf("Queue is full\n");

    return;

  }

  printf("Enter a number: ");

  scanf("%d", &x);

  if (front == -1 && rear == -1)

  {

    front = rear = 0;

    q[rear] = x;

  }

  else

  {

    rear = (rear + 1) % n;

    q[rear] = x;

  }

}

void dequeue(int \*q)

{

  if (front == -1 && rear == -1)

  {

    printf("The queue is empty\n");

    return;

  }

  else if (front == rear)

  {

    printf("The dequeued element is %d\n", q[front]);

    front = rear = -1;

  }

  else

  {

    printf("dequeued %d\n", q[front]);

    front = (front + 1) % n;

  }

}

void display(int \*q)

{

  int i = front;

  if (front == -1 && rear == -1)

  {

    printf("Queue is empty\n");

  }

  else

  {

    printf("Queue is: \n");

    while (i != rear)

    {

      printf("%d ", q[i]);

      i = (i + 1) % n;

    }

    printf("%d\n", q[rear]);

  }

}

void peek(int \*q)

{

  if (front == -1 && rear == -1)

  {

    printf("The queue is empty\n");

  }

  else

  {

    printf("The element in front is %d\n", q[front]);

  }

}

int main()

{

  int ch;

  printf("Enter the size of the queue: ");

  scanf("%d", &n);

  int \*q = (int \*)malloc(n \* sizeof(int));

  while (1)

  {

    printf("1. Enqueue\n");

    printf("2. Dequeue\n");

    printf("3. Peek\n");

    printf("4. Display\n");

    printf("5. Exit\n");

    printf("Enter your choice: ");

    scanf("%d", &ch);

    switch (ch)

    {

    case 1:

      enqueue(q);

      break;

    case 2:

      dequeue(q);

      break;

    case 3:

      peek(q);

      break;

    case 4:

      display(q);

      break;

    case 5:

      printf("Exiting...\n");

      exit(1);

    default:

      break;

    }

  }

  return 0;

}

Circular queue using linked list

#include <stdio.h>

#include <stdlib.h>

typedef struct node

{

  int data;

  struct node \*next;

} Node;

Node \*front = NULL;

Node \*rear = NULL;

void enqueue()

{

  int x;

  printf("Enter the value: ");

  scanf("%d", &x);

  Node \*newNode;

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

  newNode->data = x;

  newNode->next = NULL;

  if (rear == 0)

  {

    front = rear = newNode;

    rear->next = front;

  }

  else

  {

    rear->next = newNode;

    rear = newNode;

    rear->next = front;

  }

}

void dequeue()

{

  Node \*temp = front;

  if (front == NULL && rear == NULL)

  {

    printf("The queue is empty\n");

    return;

  }

  else if (front == rear)

  {

    printf("The last dequeued element is: %d\n", temp->data);

    front = rear = NULL;

    free(temp);

  }

  else

  {

    front = front->next;

    rear->next = front;

    printf("Dequeued: %d\n", temp->data);

    free(temp);

  }

}

void display()

{

  Node \*temp = front;

  if (front == NULL && rear == NULL)

  {

    printf("Queue is empty\n");

  }

  else

  {

    while (temp->next != front)

    {

      printf("%d -> ", temp->data);

      temp = temp->next;

    }

    printf("%d -> ", temp->data);

    temp = temp->next;

    printf("%d(f)\n", temp->data);

  }

}

void peek()

{

  if (front == NULL && rear == NULL)

  {

    printf("The queue is empty\n");

  }

  else

  {

    printf("The element in front is %d\n", front->data);

  }

}

int main()

{

  int ch;

  while (1)

  {

    printf("1. Enqueue\n");

    printf("2. Dequeue\n");

    printf("3. Peek\n");

    printf("4. Display\n");

    printf("5. Exit\n");

    printf("Enter your choice: ");

    scanf("%d", &ch);

    switch (ch)

    {

    case 1:

      enqueue();

      break;

    case 2:

      dequeue();

      break;

    case 3:

      peek();

      break;

    case 4:

      display();

      break;

    case 5:

      printf("Exiting...\n");

      exit(1);

    default:

      break;

    }

  }

  return 0;

}