**Name : Tanishq Thuse**

**Branch : SY-CS(AI)**

**Div : B**

**Roll No. : 60**

**Subject : ADS Assignment-6**

**Title : Queue Using Array and Linked List**

**Q1) Queue Implementation using Array**

**Code :**

#include<stdio.h>

#include<stdlib.h>

typedef struct Queue{

int front;

int rear;

int size;

int\* arr; // Use dynamic array

} Queue;

Queue\* Initialize(int size);

void Enqueue(int x);

int Dequeue();

void Display();

int IsEmpty();

int IsFull();

int peek();

Queue \*q;

Queue\* Initialize(int size) {

q = (Queue \*)malloc(sizeof(Queue));

q->front = q->rear = -1;

q->size = size; // Set size based on user input

q->arr = (int \*)malloc(q->size \* sizeof(int)); // Dynamically allocate memory for the array

return q;

}

void Enqueue(int x) {

if (IsFull()) {

printf("Queue is full\n");

} else {

if (q->front == -1) q->front = 0; // Move front to 0 on first enqueue

q->rear = (q->rear + 1) % q->size; // Circular increment

q->arr[q->rear] = x;

printf("%d enqueued to queue\n", x);

}

}

int Dequeue() {

int x = -1;

if (IsEmpty()) {

printf("Queue is empty\n");

} else {

x = q->arr[q->front];

printf("%d dequeued from queue\n", x);

if (q->front == q->rear) { // Queue becomes empty after dequeue

q->front = q->rear = -1;

} else {

q->front = (q->front + 1) % q->size; // Circular increment

}

}

return x;

}

int IsEmpty() {

return q->front == -1;

}

int IsFull() {

return (q->rear + 1) % q->size == q->front;

}

void Display() {

if (IsEmpty()) {

printf("Queue is empty\n");

return;

}

printf("Queue: ");

int i = q->front;

while (1) {

printf("%d ", q->arr[i]);

if (i == q->rear) break;

i = (i + 1) % q->size; // Circular increment

}

printf("\n");

}

int peek() {

if (IsEmpty()) {

printf("Queue is empty\n");

return -1;

}

return q->arr[q->front];

}

int main() {

int size, choice, value;

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

scanf("%d", &size); // User defines the size of the queue

Initialize(size); // Initialize queue with user-defined size

do {

printf("\nQueue Operations Menu:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. Peek\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1: // Enqueue

printf("Enter the value to enqueue: ");

scanf("%d", &value);

Enqueue(value);

break;

case 2: // Dequeue

Dequeue();

break;

case 3: // Display

Display();

break;

case 4: // Peek

value = peek();

if (value != -1)

printf("Peek: %d\n", value);

break;

case 5: // Exit

printf("Exiting program.\n");

break;

default:

printf("Invalid choice, please try again.\n");

}

} while (choice != 5);

// Free allocated memory

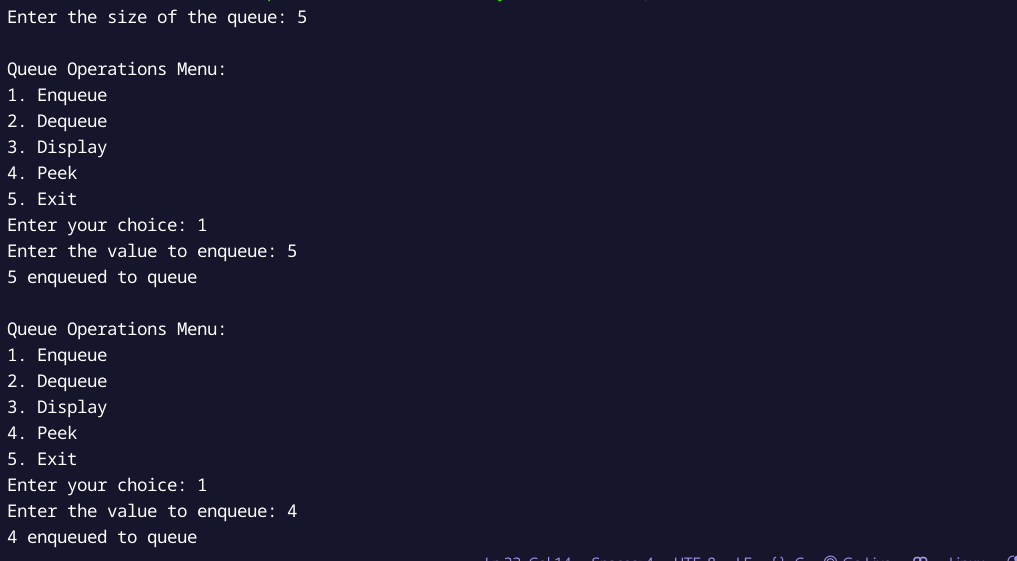
free(q->arr);

free(q);

return 0;

}

Output :



****

**Q2) Queue Implementation using Linked List**

**Code :**

#include<stdio.h>

#include<stdlib.h>

typedef struct Node{

int data;

struct Node\* next;

} Node;

typedef struct Queue{

Node\* front;

Node\* rear;

} Queue;

Queue\* Initialize();

void Enqueue(int x);

int Dequeue();

void Display();

int IsEmpty();

int peek();

Queue \*q;

Queue\* Initialize(){

q = (Queue \*)malloc(sizeof(Queue));

q->front = q->rear = NULL;

return q;

}

void Enqueue(int x){

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

if (!temp){

printf("Memory error\n");

return;

}

temp->data = x;

temp->next = NULL;

if (q->rear == NULL) {

q->front = q->rear = temp;

} else {

q->rear->next = temp;

q->rear = temp;

}

printf("%d enqueued to queue\n", x);

}

int Dequeue(){

if(IsEmpty()){

printf("Queue is empty\n");

return -1;

}

Node\* temp = q->front;

int x = temp->data;

q->front = q->front->next;

if (q->front == NULL) {

q->rear = NULL;

}

free(temp);

printf("%d dequeued from queue\n", x);

return x;

}

int IsEmpty(){

return (q->front == NULL);

}

void Display(){

if (IsEmpty()){

printf("Queue is empty\n");

return;

}

Node\* temp = q->front;

while (temp) {

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

temp = temp->next;

}

printf("\n");

}

int peek(){

if (IsEmpty()) {

printf("Queue is empty\n");

return -1;

} else {

return q->front->data;

}

}

int main(){

int choice, value;

Initialize();

do {

printf("\nQueue Operations Menu:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. Peek\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1:

printf("Enter the value to enqueue: ");

scanf("%d", &value);

Enqueue(value);

break;

case 2:

Dequeue();

break;

case 3:

Display();

break;

case 4:

value = peek();

if(value != -1)

printf("Peek: %d\n", value);

break;

case 5:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice, please try again.\n");

}

} while(choice != 5);

while (!IsEmpty()) {

Dequeue();

}

free(q);

return 0;

}

**Output :**

****