**Data Structure: Queue**

**Tutorial 6: U19CS012**

1.Write a program to implement a queue and perform basic operations of queue using array.

* + - * + En(enter)queue
        + De(lete)queue
        + peek
        + isfull
        + isempty

Code:

*// Write a program to implement a queue and perform basic operations of queue using array.*

*// •    En(enter)queue*

*// •    De(lete)queue*

*// •    peek*

*// •    isfull*

*// •    isempty*

*// •    size*

*#include* <stdio.h>

*#include* <stdlib.h>

*#define* MAX 10

*// Function Declarations*

void enqueue();

void dequeue();

*//Supportive Operations*

int peek();

int isFull();

int isEmpty();

int Size();

void display();

int Queue[MAX];

*//Initially Front and Rear Location is -1*

int front = -1;

int rear = -1;

int main()

{

*// Menu Driven C Program for Queue implementation Using Array*

    printf("\n~~~~~~Queue Operations~~~~~~\n");

    printf("1 -> Enter Element in Queue\n");

    printf("2 -> Delete Element from Queue\n");

    printf("3 -> Peek[Front] Element of Queue\n");

    printf("4 -> Check if Queue is Full or Not?\n");

    printf("5 -> Check if Queue is Empty or Not?\n");

    printf("6 -> Get the Size of Queue\n");

    printf("7 -> Display all Elements of Queue\n");

    printf("8 -> Quit\n");

    int ch;

*while* (1)

    {

        printf("Enter your choice : ");

        scanf("%d", &ch);

*switch* (ch)

        {

*case* 1:

            enqueue();

*break*;

*case* 2:

            dequeue();

*break*;

*case* 3:;

            int peek\_ele = peek();

            printf("The Front[Peek] Element of Queue is : %d\n", peek\_ele);

*break*;

*case* 4:

*if* (isFull())

            {

                printf("Queue is Full!\n");

            }

*else*

            {

                printf("Queue is Not Full!\n");

            }

*break*;

*case* 5:

*if* (isEmpty())

            {

                printf("Queue is Empty!\n");

            }

*else*

            {

                printf("Queue is Not Empty!\n");

            }

*break*;

*case* 6:;

            int sz;

            sz = Size();

            printf("Size of Queue : %d\n", sz);

*break*;

*case* 7:

            display();

*break*;

*case* 8:

            exit(1);

*default*:

            printf("Enter Valid Choice!\n");

        }

    }

}

*// Function Definations*

int isEmpty()

{

*if* (front == -1 || front == rear + 1)

*return* 1;

*else*

*return* 0;

}

int isFull()

{

*if* (rear == MAX - 1)

*return* 1;

*else*

*return* 0;

}

int peek()

{

*if* (isEmpty())

    {

        printf("Queue Underflow!\n");

        exit(1);

    }

*return* Queue[front];

}

void enqueue()

{

*if* (isFull())

        printf("Queue Overflow!\n");

*else*

    {

*if* (front == -1)

            front = 0;

        int ele;

        printf("Enter the Element to Insert in Queue : ");

        scanf("%d", &ele);

        rear = rear + 1;   *// Increment Rear*

        Queue[rear] = ele; *// Assign Value at Back*

    }

}

void dequeue()

{

*if* (isEmpty())

    {

        printf("Queue Underflow!\n");

*return*;

    }

*else*

    {

        printf("Element Deleted from Queue is : %d\n", Queue[front]);

        front = front + 1;

    }

}

int Size()

{

*if* (isEmpty())

    {

        printf("Queue is Empty!\n");

*return* 0;

    }

*else*

    {

*return* rear - front + 1;

    }

}

void display()

{

*if* (isEmpty())

        printf("Queue is Empty!\n");

*else*

    {

        int i;

        printf("Queue is :\n");

*for* (i = front; i < rear; i++)

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

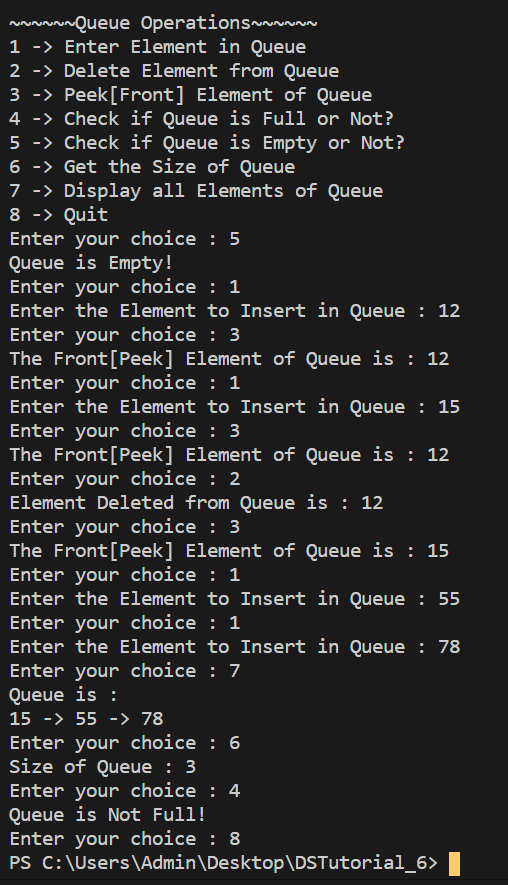
*// Rear Element*

        printf("%d\n", Queue[i]);

    }

}

Example Test Cases:



2. Perform above operations to implement DEQueue (double ended queue)

Code:

*#include* <stdio.h>

*#include* <conio.h>

*#define* SIZE 100

*// Function Defination*

void enQueue();

*// For INPUT Restricted Queue*

int deQueueFront();

int deQueueRear();

*// For OUTPUT Restricted Queue*

void enQueueRear();

void enQueueFront();

void display();

int queue[SIZE];

int rear = 0, front = 0;

int main()

{

    char ch;

    int ch1, ch2, value = 0;

    printf("\n~~~~~~~ Type of Double Ended Queue ~~~~~~~\n");

*do*

    {

        printf("\n1 -> Input-restricted deque \n");

        printf("2 -> Output-restricted deque \n");

        printf("\nEnter your choice of Queue Type : ");

        scanf("%d", &ch1);

*switch* (ch1)

        {

*case* 1:

            printf("\n~~~ Operations For Input Restricted Deque ~~~\n");

            printf("1 -> Insert\n2 -> Delete from Rear\n3 -> Delete from Front\n4 ->  Display");

*do*

            {

                printf("\nEnter Choice for DeQueue Operation : ");

                scanf("%d", &ch2);

*switch* (ch2)

                {

*case* 1:

                    enQueueRear();

*// display();*

*break*;

*case* 2:

                    value = deQueueRear();

*if* (value)

                        printf("\nThe value deleted is %d", value);

*// display();*

*break*;

*case* 3:

                    value = deQueueFront();

*if* (value)

                        printf("\nThe value deleted is %d", value);

*// display();*

*break*;

*case* 4:

                    display();

*break*;

*default*:

                    printf("Wrong choice");

                }

                printf("\nDo you want to perform another operation (Y/N): ");

                ch = getch();

            } *while* (ch == 'y' || ch == 'Y');

            getch();

*break*;

*case* 2:

            printf("\n~~~ Operations For Output Restricted Deque ~~~\n");

            printf("1 -> Insert at Rear\n2 -> Insert at Front\n3 -> Delete\n4 -> Display");

*do*

            {

                printf("\nEnter your choice for the operation: ");

                scanf("%d", &ch2);

*switch* (ch2)

                {

*case* 1:

                    enQueueRear();

*// display();*

*break*;

*case* 2:

                    enQueueFront();

*// display();*

*break*;

*case* 3:

                    value = deQueueFront();

*if* (value)

                        printf("\nThe value deleted is %d", value);

*// display();*

*break*;

*case* 4:

                    display();

*break*;

*default*:

                    printf("Wrong choice");

                }

                printf("\nDo you want to perform another operation (Y/N): ");

                ch = getch();

            } *while* (ch == 'y' || ch == 'Y');

            getch();

*break*;

        }

        printf("\nDo you want to continue with Another Deque(y/n):");

        ch = getch();

    } *while* (ch == 'y' || ch == 'Y');

}

void enQueueRear()

{

    char ch;

*if* (front == SIZE / 2)

    {

        printf("\nQueue is full!!! Insertion is not possible!!! ");

*return*;

    }

*do*

    {

        int value;

        printf("\nEnter Value to be Inserted:");

        scanf("%d", &value);

        queue[front] = value;

        front++;

        printf("Continue Insertion(Y/N)?");

        ch = getch();

    } *while* (ch == 'y' || ch == 'Y');

}

void enQueueFront()

{

    char ch;

*if* (front == SIZE / 2)

    {

        printf("\nQueue is full! Insertion Not Possible!");

*return*;

    }

*do*

    {

        int value;

        printf("\nEnter Value to be Inserted:");

        scanf("%d", &value);

        rear--;

        queue[rear] = value;

        printf("Continue Insertion(Y/N)?");

        ch = getch();

    } *while* (ch == 'y' || ch == 'Y');

}

int deQueueRear()

{

    int deleted;

*if* (front == rear)

    {

        printf("\nQueue is Empty! Deletion Not Possible!");

*return* 0;

    }

    front--;

    deleted = queue[front + 1];

*return* deleted;

}

int deQueueFront()

{

    int deleted;

*if* (front == rear)

    {

        printf("\nQueue is Empty! Deletion Not Possible!");

*return* 0;

    }

    rear++;

    deleted = queue[rear - 1];

*return* deleted;

}

void display()

{

    int i;

*if* (front == rear)

        printf("\nQueue is Empty!");

*else*

    {

        printf("\nThe Queue Elements are : ");

*for* (i = rear; i < front - 1; i++)

        {

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

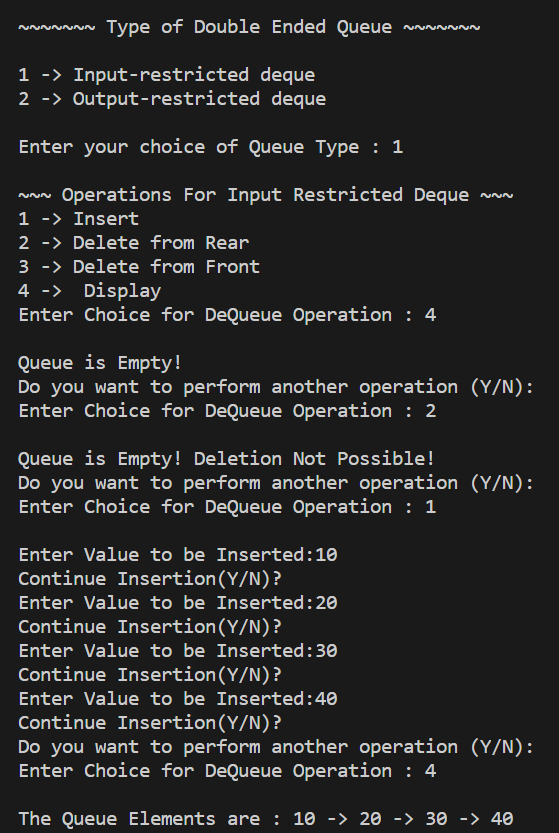
        }

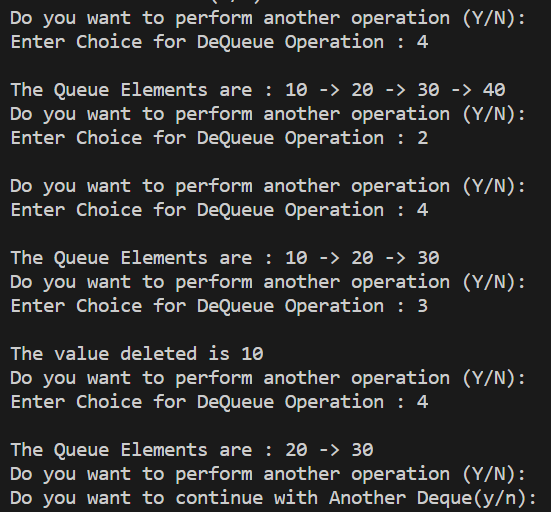
        printf("%d", queue[i]);

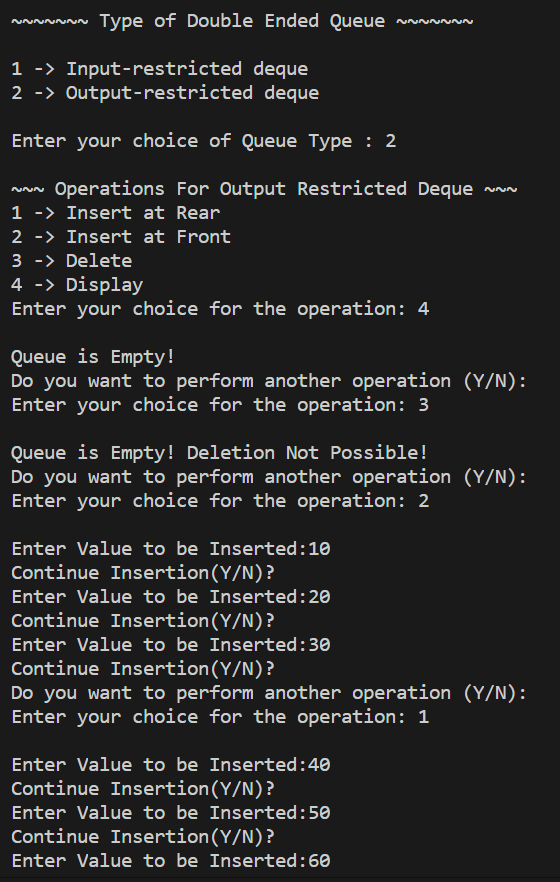
    }

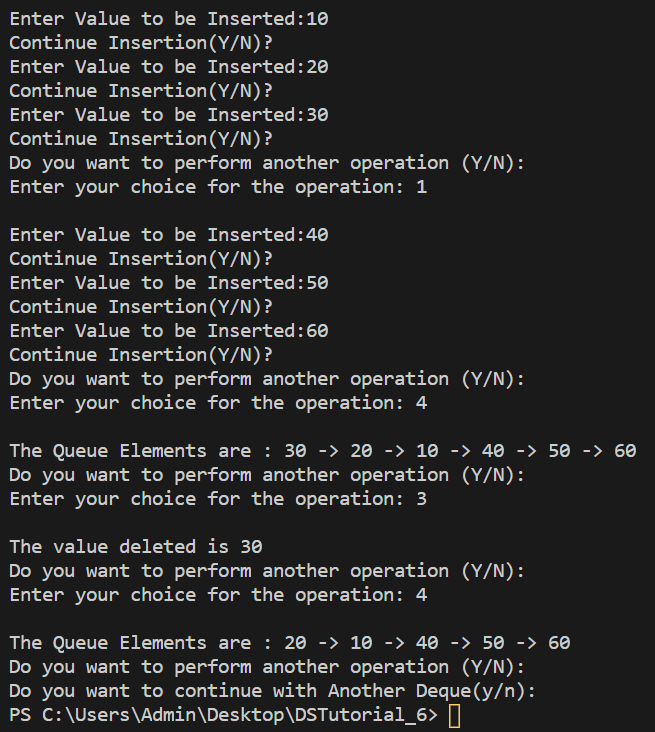
}

Example Test Cases:









3. Perform above operations to implement Priority Queue

Code:

*#include* <stdio.h>

*#define* SIZE 100

*// To Store Data Element*

int Priority\_Queue[SIZE];

*// To Store Priority of Data Element*

int Priority[SIZE];

*// rear -> r & front -> f*

int r = -1, f = -1;

*//enqueuePQ function -> Insert Data and its Priority in queue*

void enqueuePQ(int data, int p);

*//Display Priority Queue*

void displayPQ();

*//Delete Element from Front of Priority Queue*

int dequeuePQ();

int main()

{

    int choice, n, i, data, p;

*do*

    {

        printf("\n1 -> Insert the Data in Priority Queue");

        printf("\n2 -> Display Priority Queue");

        printf("\n3 -> Delete the data from the Priority Queue");

        printf("\n0 -> Exit\n");

        printf("Enter Your Choice : ");

        scanf("%d", &choice);

*switch* (choice)

        {

*case* 1:

            printf("\nEnter the Number of Data to be Inserted : ");

            scanf("%d", &n);

            printf("\nEnter your Data and Priority of Data :\n");

            i = 0;

*while* (i < n)

            {

                scanf("%d %d", &data, &p);

                enqueuePQ(data, p);

                i++;

            }

*break*;

*case* 2:

            displayPQ();

*break*;

*case* 3:

            dequeuePQ();

*break*;

*case* 0:

*break*;

*default*:

            printf("\nIncorrect Choice Entered!");

        }

    } *while* (choice != 0);

*return* 0;

}

*//enqueuePQ function to insert data and its priority in queue*

void enqueuePQ(int data, int p)

{

    int i;

*//Check if Queue is full*

*if* ((f == 0) && (r == SIZE - 1))

        printf("Queue is Full!");

*else*

    {

*//if Queue is empty*

*if* (f == -1)

        {

            f = r = 0;

            Priority\_Queue[r] = data;

            Priority[r] = p;

        }

*else* *if* (r == SIZE - 1)

        {

*//if there there is some elemets in Queue*

*// Insert Element at Right Priority Position*

*for* (i = f; i <= r; i++)

            {

                Priority\_Queue[i - f] = Priority\_Queue[i];

                Priority[i - f] = Priority[i];

                r = r - f;

                f = 0;

*for* (i = r; i > f; i--)

                {

*if* (p > Priority[i])

                    {

                        Priority\_Queue[i + 1] = Priority\_Queue[i];

                        Priority[i + 1] = Priority[i];

                    }

*else*

*break*;

                    Priority\_Queue[i + 1] = data;

                    Priority[i + 1] = p;

                    r++;

                }

            }

        }

*else*

        {

*for* (i = r; i >= f; i--)

            {

*if* (p > Priority[i])

                {

                    Priority\_Queue[i + 1] = Priority\_Queue[i];

                    Priority[i + 1] = Priority[i];

                }

*else*

*break*;

            }

            Priority\_Queue[i + 1] = data;

            Priority[i + 1] = p;

            r++;

        }

    }

}

*//Display Priority Queue*

void displayPQ()

{

    int i;

*if* (f == -1)

    {

        printf("Queue is Empty!");

*return*;

    }

    printf("Priority Queue [Element,Priority] : \n\n");

*for* (i = f; i <= r; i++)

    {

        printf("[ %d , %d ]\n", Priority\_Queue[i], Priority[i]);

    }

}

*//Delete Element from Front of Priority Queue*

int dequeuePQ()

{

*if* (f == -1)

    {

        printf("Queue is Empty!");

    }

*else*

    {

        printf("Deleted Data [Element,Priority] = [ %d , %d ]\n", Priority\_Queue[f], Priority[f]);

*if* (f == r)

            f = r = -1;

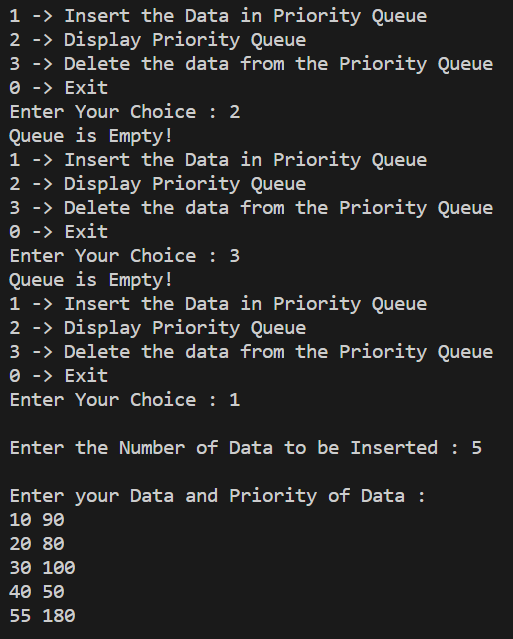
*else*

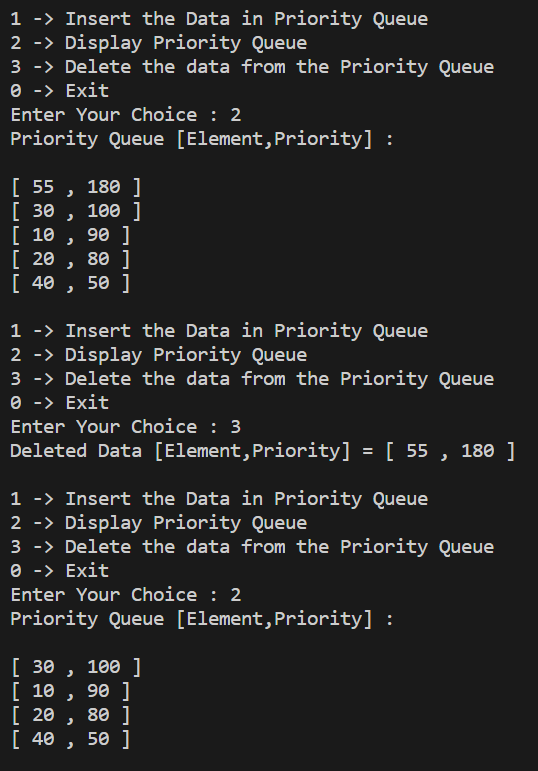
            f++;

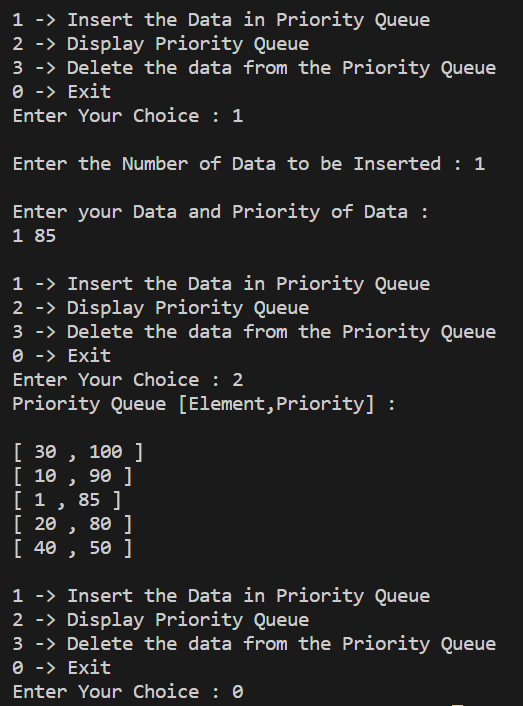
    }

}

Example Test Cases:







*We have* ***successfully Implemented*** *and* ***Verified*** *Queue, Deque and Priority Queue in C.*

Submitted By:

Roll Number: **U19CS012** (*D-12*)

Name: *Bhagya Rana*