**Batch: C3 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:** Implementation of Queue operations (Static and Dynamic implementation)- Queue,  circular queue, priority queue, and deque |

**Objective:** To implement Basic Operations of Queues

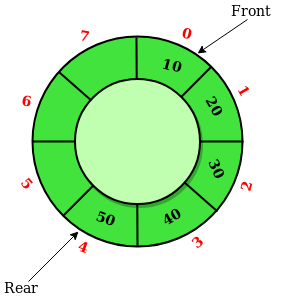
**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **2** | Apply linear and non-linear data structure in application development. |

**Books/ Journals/ Websites referred:**

**Introduction:**

Diagram of circular Queue dynamic



**Program source code:**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct node{**

**int data;**

**struct node \*link;**

**};**

**struct node \*front = NULL;**

**struct node \*rear = NULL;**

**void enqueue(int n){**

**struct node \*newnode;**

**newnode = (struct node \*)malloc(sizeof(struct node));**

**newnode-> data = n;**

**newnode-> link = NULL;**

**if(front == NULL && rear == NULL){**

**front = rear = newnode;**

**rear-> link = front;**

**}**

**else{**

**rear ->link = newnode;**

**rear = newnode;**

**rear->link = front;**

**}**

**}**

**void dequeue(){**

**struct node \*temp;**

**temp = front;**

**if(front == NULL && rear == NULL){**

**printf("Queue is empty.\n");**

**}**

**else if(front == rear){**

**front = rear = NULL;**

**free(temp);**

**}**

**else{**

**front = front->link;**

**rear->link = front;**

**free(temp);**

**}**

**}**

**void display(){**

**struct node \*temp;**

**temp = front;**

**if(front == NULL && rear == NULL){**

**printf("The queue is empty.\n");**

**}**

**else{**

**while(temp->link != front){**

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

**temp = temp->link;**

**}**

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

**printf("\n");**

**}**

**}**

**int main(){**

**int choice, value;**

**while(1){**

**printf("\nMenu:\n");**

**printf("1. Enqueue\n");**

**printf("2. Dequeue\n");**

**printf("3. Display\n");**

**printf("4. 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:**

**printf("Exiting program...");**

**exit(0);**

**default:**

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

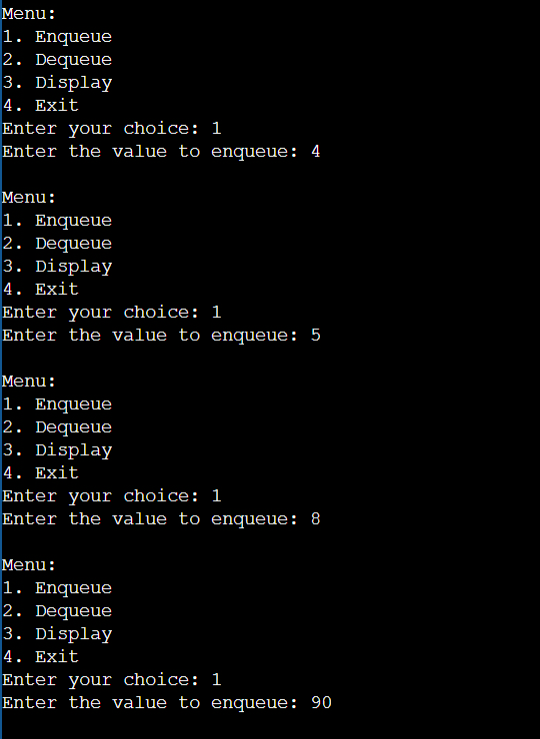
**}**

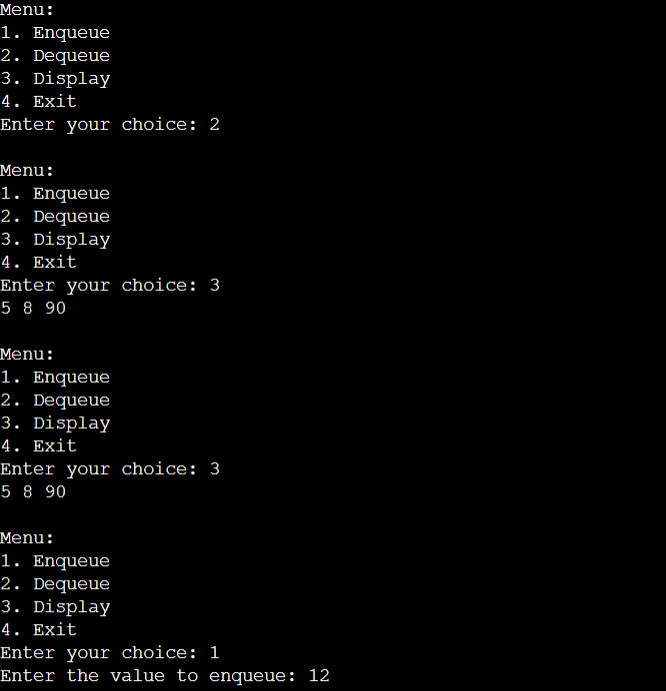
**}**

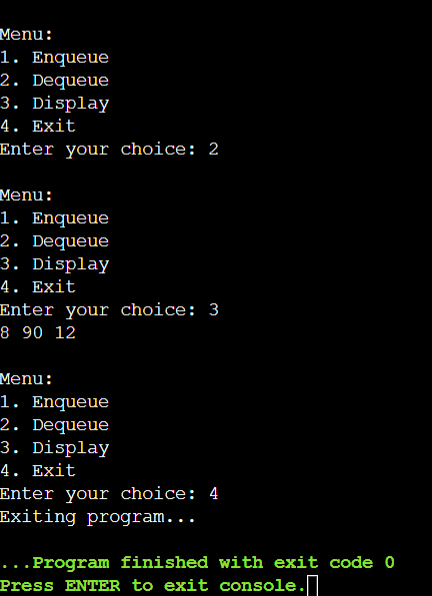
**return 0;**

**}**

**Output Screenshots:**







**Conclusion:-**

By doing this experiment we learnt how to implement circular queue using dynamic memory allocation and other types of queue using static and dynamic memory allocation.

**Post lab questions:**

1. What are the key considerations in choosing between a circular array and a linked list for the implementation of a queue?

Ans. The key consideration between choosing a circular array or a linked list for the implementation of a queue memory usage. Circular array pre-allocates a fixed amount of memory while linked list allocates it dynamically.