**DS LAB 4**

**NAME :** Vivian Vijay Ludrick

**BRANCH :** SE Comps-A Batch -C

**ROLL NO: 9914**

**1. Static Implementation of Queue of integers.**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

*// creating a structure for queue to store the float.*

typedef struct

{

int a[SIZE];

int front, rear;

} Queue;

*// adding elements to the queue*

void enqueue(Queue \**p*, int *ele*)

{

*// if the queue is full, then*

if (*p*->rear == SIZE - 1)

{

printf("\nQueue Full\n");

printf("----------------------------------------------------------------");

return;

}

*p*->rear++;

*p*->a[*p*->rear] = *ele*;

*// front should only be updated when size is -1 and when elements are dequeued*

if (*p*->front == -1)

*p*->front = 0;

printf("Element enqueued is %d\n", *ele*);

printf("----------------------------------------------------------------");

}

*// displays the elements of the queue*

void display(Queue *q*)

{

printf("\nElements are:\n");

for (int i = *q*.front; i <= *q*.rear; i++)

{

printf("\t%d\n", *q*.a[i]);

}

printf("----------------------------------------------------------------");

}

*// remove the first element in the queue and returns it*

int dequeue(Queue \**p*)

{

if (*p*->front == -1)

{

printf("\nQueue Empty");

return -1;

}

int ele = *p*->a[*p*->front];

if (*p*->front == *p*->rear)

{

*p*->front = *p*->rear = -1;

printf("The queue has been reset as there are no more elements in the queue.\n");

*// to prevent reseting we can put a flag which flip for this condition and then check if the flag is flipped or not for the enqueue and dequeue method.Or we can just ignore this condition as this condition is used to reset the queue*

}

else

{

*p*->front += 1;

}

return ele;

}

*// returns the size of the queue*

int getSize(Queue \**q*)

{

int size = *q*->rear - *q*->front + 1;

printf("\nSize is %d\n", size);

printf("----------------------------------------------------------------");

return size;

}

*// checks whether the size is empty or not*

void isEmpty(Queue \**q*)

{

if (*q*->front == -1)

{

printf("\nQueue is empty\n");

}

else

{

printf("\nNot Empty\n");

}

printf("----------------------------------------------------------------");

}

*// check whether the queue is full or not*

void isFull(Queue \**q*)

{

if (*q*->rear == SIZE - 1)

{

printf("\nFull\n");

}

else

{

printf("\nNot Full\n");

}

printf("----------------------------------------------------------------");

}

*// start of the main function*

int main()

{

Queue q1;

q1.front = q1.rear = -1; *// initializing an empty queue*

int choice, ele;

do

{

printf("\nMenu:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. Check Empty\n");

printf("5. Check Full\n");

printf("6. Get Size\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

printf("----------------------------------------------------------------");

switch (choice)

{

case 1:

printf("\nEnter element to enqueue: ");

scanf("%d", &*ele*);

enqueue(&*q1*, ele);

break;

case 2:

ele = dequeue(&*q1*);

if (ele != -1)

printf("\nRemoved element is: %d\n", ele);

printf("----------------------------------------------------------------");

break;

case 3:

display(q1);

break;

case 4:

isEmpty(&*q1*);

break;

case 5:

isFull(&*q1*);

break;

case 6:

getSize(&*q1*);

break;

case 7:

printf("\nExiting program...\n");

exit(0);

default:

printf("\nInvalid choice.\n");

printf("----------------------------------------------------------------");

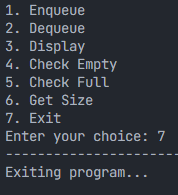
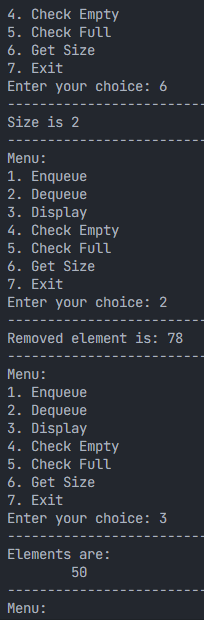
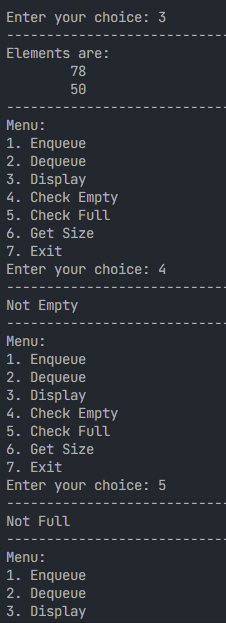
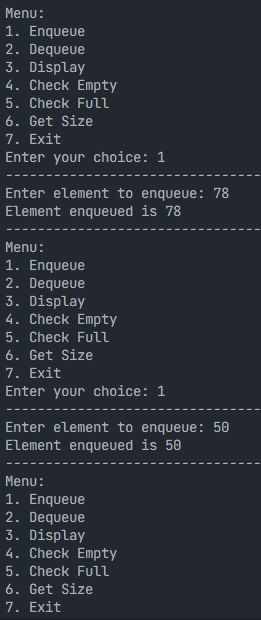
}

} while (choice != 7);

return 0;

}

**OUTPUT:**

****

**Q.2.WAP to implement Queue of Students.**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

#define STRING\_SIZE 50

*// struct of a student*

typedef struct

{

char name[STRING\_SIZE];

int rollNo;

} Student;

*// struct of a queue of students*

typedef struct

{

Student a[SIZE];

int front, rear;

} Queue;

*// enter the student details in a queue*

void enqueue(Queue \**p*, Student *s1*)

{

if (*p*->rear == SIZE - 1)

{

printf("\nQueue Full");

return;

}

*p*->rear++;

*p*->a[*p*->rear] = *s1*;

if (*p*->front == -1)

*p*->front = 0;

printf("\nElement enqueued is Student: %s and roll no %d", *s1*.name, *s1*.rollNo);

}

*// prints the details of the student that are in queue*

void display(Queue *q*)

{

printf("\nElements are:");

for (int i = *q*.front; i <= *q*.rear; i++)

{

printf("\nStudent: %s and Roll no: %d", *q*.a[i].name, *q*.a[i].rollNo);

}

}

*// removes the first student from the queue*

Student dequeue(Queue \**p*)

{

if (*p*->front == -1)

{

printf("\nQueue Empty");

}

Student ele = *p*->a[*p*->front];

if (*p*->front == *p*->rear)

*p*->front = *p*->rear = -1;

else

*p*->front += 1;

return ele;

}

*// returns the number of students in the queue*

int getSize(Queue \**q*)

{

int size = *q*->rear - *q*->front + 1;

printf("\nSize is %d", size);

return size;

}

*// checks whether the queue is empty or not*

void isEmpty(Queue \**q*)

{

if (*q*->front == -1)

{

printf("\nQueue is empty");

}

else

{

printf("\nNot Empty");

}

}

*// checks whether the queue is full or not*

void isFull(Queue \**q*)

{

if (*q*->rear == SIZE - 1)

{

printf("\nFull");

}

else

{

printf("\nNot Full");

}

}

*// start of the main functions*

int main()

{

Queue q1;

q1.front = q1.rear = -1;

int choice;

Student s1;

do

{

printf("\nMenu:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. Check Empty\n");

printf("5. Check Full\n");

printf("6. Get Size\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("Enter student name: ");

scanf(" %[^\n]", s1.name);

printf("Enter student roll number: ");

scanf("%d", &s1.rollNo);

enqueue(&*q1*, s1);

break;

case 2:

s1 = dequeue(&*q1*);

if (s1.rollNo != -1)

{

printf("\nRemoved student: %s with Roll no: %d", s1.name, s1.rollNo);

}

break;

case 3:

display(q1);

break;

case 4:

isEmpty(&*q1*);

break;

case 5:

isFull(&*q1*);

break;

case 6:

getSize(&*q1*);

break;

case 7:

printf("Exiting program.\n");

exit(0);

break;

default:

printf("Invalid choice.\n");

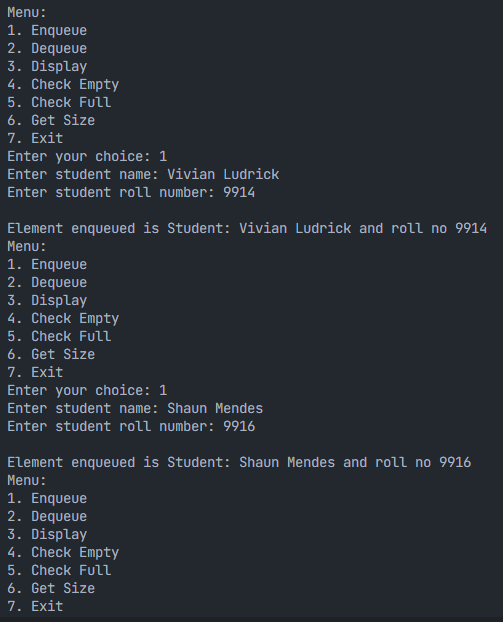
}

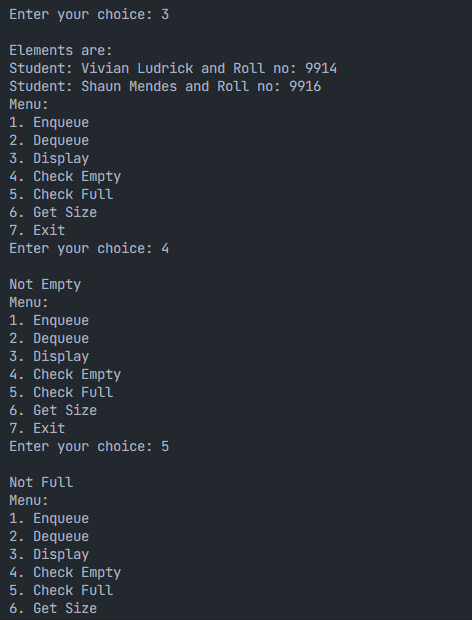
} while (choice != 7);

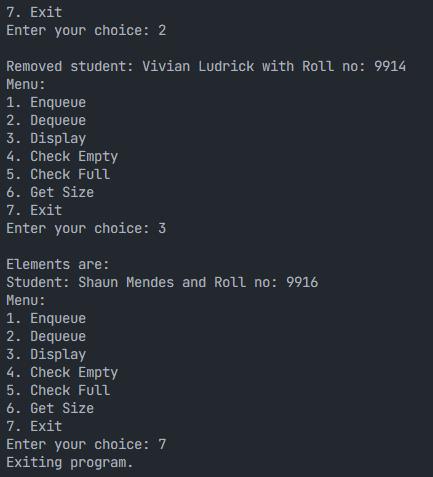
return 0;

}

**OUTPUT:**

****

****

****