AM.EN.U4AIE22017

DEVAPRIYA S

LAB 6 -QUEUE

**Question 1:**

Design a menu driven program to implement a queue using a dynamic array. Include the options for enqueue, dequeue and display operations

#*include* <stdio.h>

#*include* <stdlib.h>

#*define* *MAX\_SIZE* 100

int\* *createQueue*() {

    int\* queue = (int\*)*malloc*(*MAX\_SIZE* \* sizeof(int));

*return* queue;

}

int *isFull*(int front, int rear) {

*return* ((rear + 1) % *MAX\_SIZE* == front);

}

int *isEmpty*(int front, int rear) {

*return* (front == -1);

}

void *enqueue*(int\* queue, int\* front, int\* rear, int data) {

*if* (*isFull*(\*front, \*rear)) {

*printf*("Queue is full. Cannot enqueue.\n");

*return*;

    }

*if* (\*front == -1)

        \*front = 0;

    \*rear = (\*rear + 1) % *MAX\_SIZE*;

    queue[\*rear] = data;

*printf*("Enqueued element: %d\n", data);

}

int *dequeue*(int\* queue, int\* front, int\* rear) {

*if* (*isEmpty*(\*front, \*rear)) {

*printf*("Queue is empty. Cannot dequeue.\n");

*return* -1;

    }

    int data = queue[\*front];

*if* (\*front == \*rear)

        \*front = \*rear = -1;

*else*

        \*front = (\*front + 1) % *MAX\_SIZE*;

*printf*("Dequeued element: %d\n", data);

*return* data;

}

void *display*(int\* queue, int front, int rear) {

*if* (*isEmpty*(front, rear)) {

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

*return*;

    }

*printf*("Queue elements: ");

    int i = front;

*while* (i != rear) {

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

        i = (i + 1) % *MAX\_SIZE*;

    }

*printf*("%d\n", queue[rear]);

}

void *destroyQueue*(int\* queue) {

*free*(queue);

}

int *main*() {

    int\* queue = *createQueue*();

    int front = -1;

    int rear = -1;

    int choice, data;

*while* (1) {

*printf*("\nChoose the queue operation:\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 element to enqueue: ");

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

*enqueue*(queue, &front, &rear, data);

*break*;

*case* 2:

*dequeue*(queue, &front, &rear);

*break*;

*case* 3:

*display*(queue, front, rear);

*break*;

*case* 4:

*destroyQueue*(queue);

*printf*("Well then, bye and have a nice day\n");

*return* 0;

*default*:

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

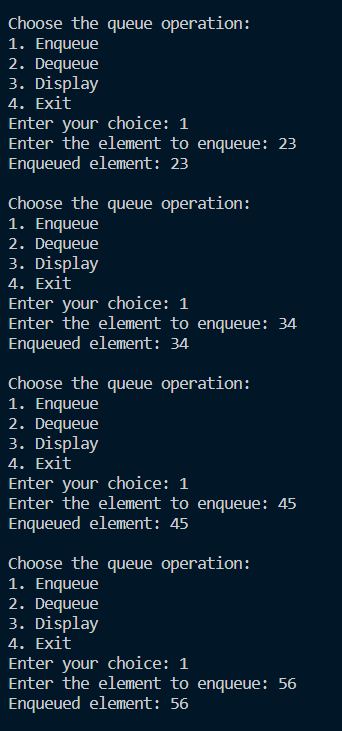
        }

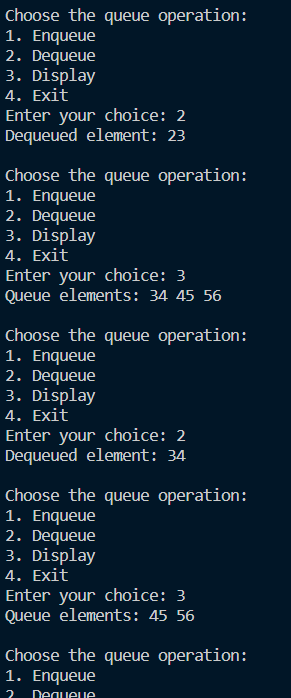
    }

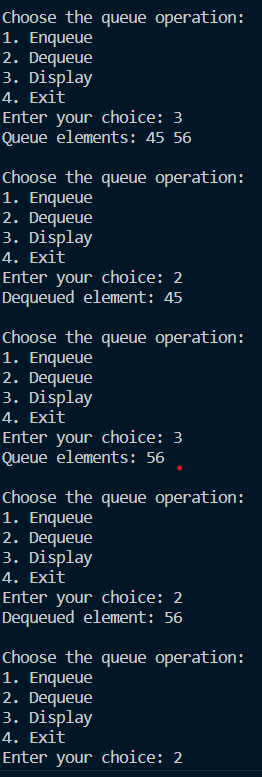
*return* 0;

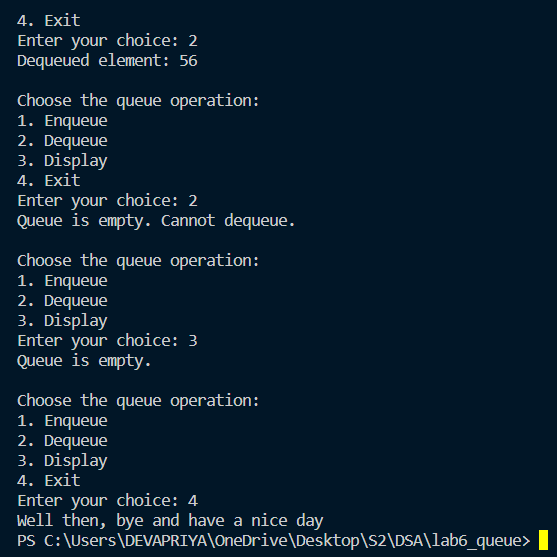
}

Output:









**Question 2:**

Design a menu driven program to implement a circular queue using a dynamic array. Include the options for enqueue, dequeue and display operations

#*include* <stdio.h>

#*include* <stdlib.h>

#*define* *MAX\_SIZE* 100

int\* *createQueue*(int maxSize) {

    int\* queue = (int\*)*malloc*(maxSize \* sizeof(int));

*return* queue;

}

int *isFull*(int front, int rear, int size) {

*return* ((rear + 1) % size == front);

}

int *isEmpty*(int front, int rear) {

*return* (front == -1);

}

void *enqueue*(int\* queue, int\* front, int\* rear, int size, int data) {

*if* (*isFull*(\*front, \*rear, size)) {

*printf*("Queue is full. Cannot enqueue.\n");

*return*;

    }

*if* (*isEmpty*(\*front, \*rear))

        \*front = 0;

    \*rear = (\*rear + 1) % size;

    queue[\*rear] = data;

*printf*("Enqueued element: %d\n", data);

}

int *dequeue*(int\* queue, int\* front, int\* rear, int size) {

*if* (*isEmpty*(\*front, \*rear)) {

*printf*("Queue is empty. Cannot dequeue.\n");

*return* -1;

    }

    int data = queue[\*front];

*if* (\*front == \*rear)

        \*front = \*rear = -1;

*else*

        \*front = (\*front + 1) % size;

*printf*("Dequeued element: %d\n", data);

*return* data;

}

void *display*(int\* queue, int front, int rear, int size) {

*if* (*isEmpty*(front, rear)) {

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

*return*;

    }

*printf*("Queue elements: ");

    int i = front;

*do* {

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

        i = (i + 1) % size;

    } *while* (i != (rear + 1) % size);

*printf*("\n");

}

void *destroyQueue*(int\* queue) {

*free*(queue);

}

int *main*() {

    int maxSize, choice, data;

    int\* queue;

    int front = -1;

    int rear = -1;

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

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

    queue = *createQueue*(maxSize);

*while* (1) {

*printf*("\nChoose the queue Operation\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 element to enqueue: ");

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

*enqueue*(queue, &front, &rear, maxSize, data);

*break*;

*case* 2:

*dequeue*(queue, &front, &rear, maxSize);

*break*;

*case* 3:

*display*(queue, front, rear, maxSize);

*break*;

*case* 4:

*destroyQueue*(queue);

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

*return* 0;

*default*:

*printf*("Invalid choice!!\n");

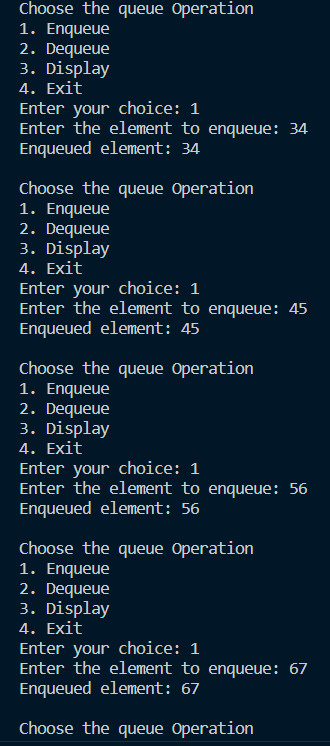
        }

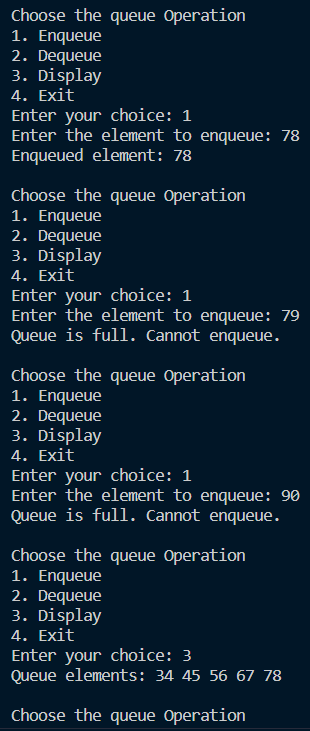
    }

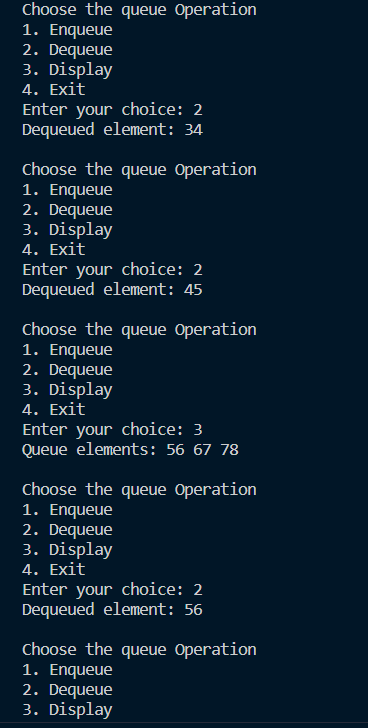
*return* 0;

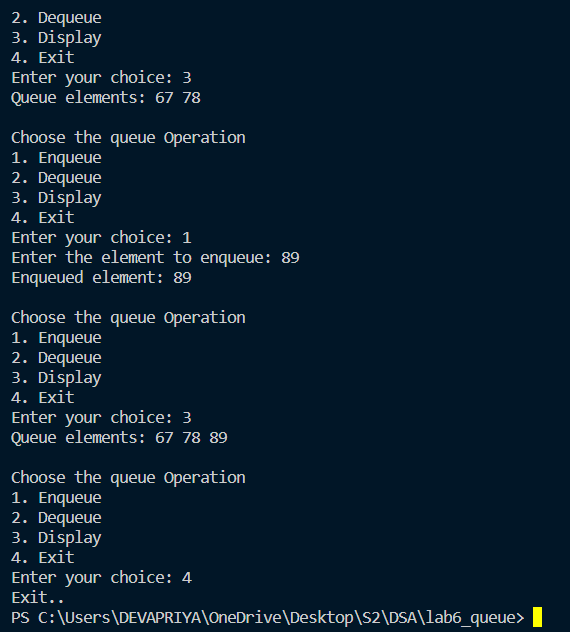
}

Output:









**Question 3:**

Implement the following methods in the above circular queue: (a) splitq(), to split a queue into two queues so that all items in odd positions are in one queue and those in even positions are in another queue. (b) getminElement () to return the minimum element in a queue.

#*include* <stdio.h>

#*include* <stdlib.h>

#*define* *MAX* 10

int queue[*MAX*];

int front = -1, rear = -1;

void *enqueue*(int item) {

*if* ((front == 0 && rear == *MAX* - 1) || (front == rear + 1)) {

*printf*("Queue is full\n");

    } *else* {

*if* (front == -1) {

            front = 0;

        }

        rear = (rear + 1) % *MAX*;

        queue[rear] = item;

    }

}

int *dequeue*() {

    int item;

*if* (front == -1) {

*return* -1;

    } *else* {

        item = queue[front];

*if* (front == rear) {

            front = -1;

            rear = -1;

        } *else* {

            front = (front + 1) % *MAX*;

        }

*return* item;

    }

}

void *display*() {

    int i;

*if* (front == -1) {

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

    } *else* {

*printf*("The elements in the queue are:\n");

*for* (i = front; i != rear; i = (i + 1) % *MAX*) {

*printf*("%d\n", queue[i]);

        }

*printf*("%d\n", queue[i]);

    }

}

void *splitq*() {

    int i, j = 0, k = 0;

    int queue1[*MAX*], queue2[*MAX*];

*if* (front == -1) {

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

    } *else* {

*for* (i = front; i != rear; i = (i + 1) % *MAX*) {

*if* (i % 2 == 0) {

                queue1[j++] = queue[i];

            } *else* {

                queue2[k++] = queue[i];

            }

        }

*if* (i % 2 == 0) {

            queue1[j++] = queue[i];

        } *else* {

            queue2[k++] = queue[i];

        }

*printf*("The elements in the first queue are:\n");

*for* (i = 0; i < j; i++) {

*printf*("%d\n", queue1[i]);

        }

*printf*("The elements in the second queue are:\n");

*for* (i = 0; i < k; i++) {

*printf*("%d\n", queue2[i]);

        }

    }

}

int *getminElement*() {

    int i, min;

*if* (front == -1) {

*return* -1;

    } *else* {

        min = queue[front];

*for* (i = front; i != rear; i = (i + 1) % *MAX*) {

*if* (queue[i] < min) {

                min = queue[i];

            }

        }

*if* (queue[i] < min) {

            min = queue[i];

        }

*return* min;

    }

}

int *main*() {

    int choice, item;

*while* (1) {

*printf*("\nChoose your Queue Operation\n");

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

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

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

*printf*("4. Split\n");

*printf*("5. Get Minimum Element\n");

*printf*("6. Exit\n");

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

*switch* (choice) {

*case* 1:

*printf*("Enter the element to be inserted: ");

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

*enqueue*(item);

*display*();

*break*;

*case* 2:

                item = *dequeue*();

*if* (item == -1) {

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

                } *else* {

*printf*("The deleted element is %d\n", item);

                }

*break*;

*case* 3:

*display*();

*break*;

*case* 4:

*splitq*();

*break*;

*case* 5:

                item = *getminElement*();

*if* (item == -1) {

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

                } *else* {

*printf*("The minimum element is %d\n", item);

                }

*break*;

*case* 6:

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

*exit*(0);

*default*:

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

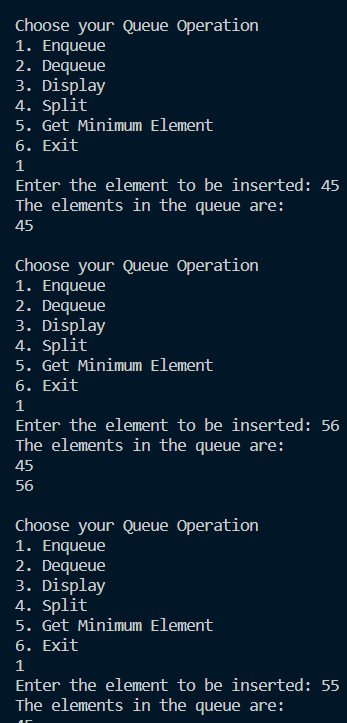
        }

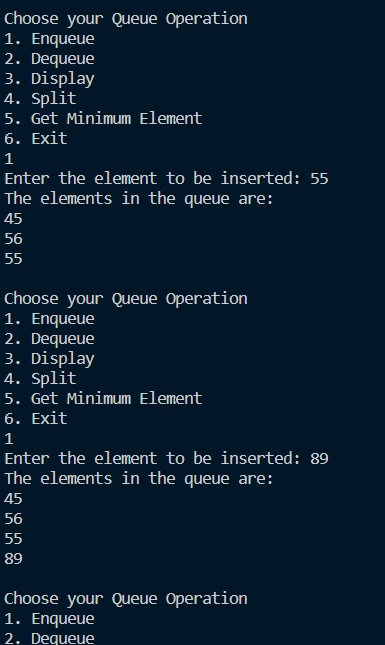
    }

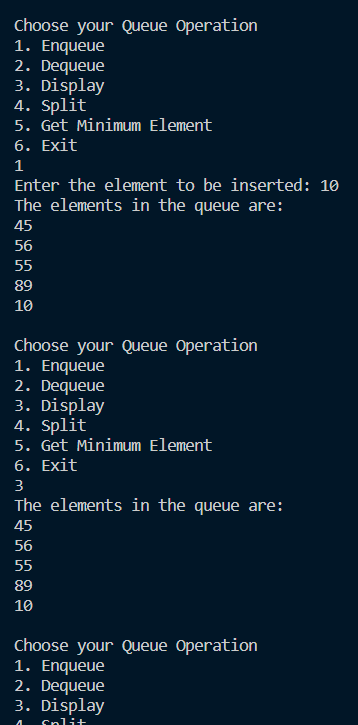
*return* 0;

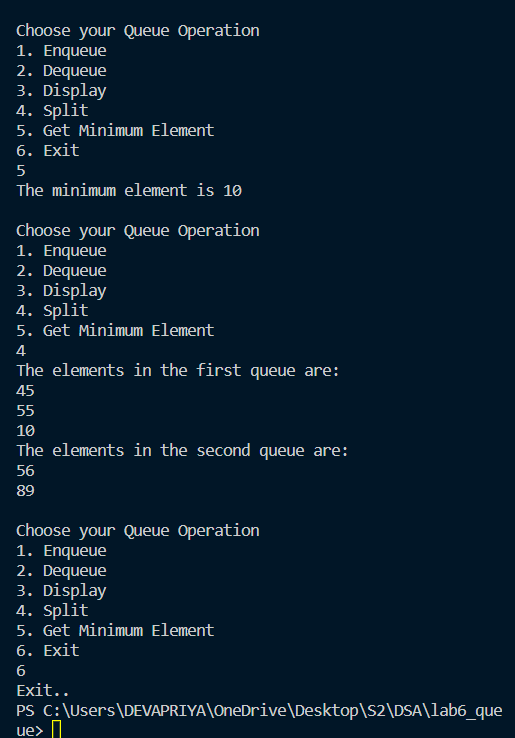
}

Output:









**Question 4:**

Implement the following operations on Deque(Double Ended Queue) using a circular array.

(a) insertFront(): Adds an item at the front of Deque. (b) insertLast(): Adds an item at the rear of Deque.

(c) deleteFront(): Deletes an item from front of Deque. (d) deleteLast(): Deletes an item from rear of Deque.

(e) getFront(): Gets the front item from queue. (f) getRear(): Gets the last item from queue.

(g) isEmpty(): Checks whether Deque is empty or not. (h) isFull(): Checks whether Deque is full or not.

(i) display(): Display queue elements starting from front to rear Test cases:

Create a queue of size 5 inset

Front1(10): insertLast(20): insetFront(30): deleteFront(): deleteLast(): insertLast(25): insetFront(40): insetFront(50): getRear(): getFront():

#*include* <stdio.h>

#*include* <stdlib.h>

#*define* *MAX\_SIZE* 5

int deque[*MAX\_SIZE*];

int front = -1, rear = -1;

int *isFull*() {

*return* ((front == 0 && rear == *MAX\_SIZE* - 1) || (front == rear + 1));

}

int *isEmpty*() {

*return* (front == -1);

}

void *insertFront*(int item) {

*if* (*isFull*()) {

*printf*("Deque is full. Cannot insert at the front.\n");

    } *else* {

*if* (front == -1) {

            front = 0;

            rear = 0;

        } *else* *if* (front == 0) {

            front = *MAX\_SIZE* - 1;

        } *else* {

            front = front - 1;

        }

        deque[front] = item;

*printf*("Inserted %d at the front of the Deque.\n", item);

    }

}

void *insertLast*(int item) {

*if* (*isFull*()) {

*printf*("Deque is full. Cannot insert at the rear.\n");

    } *else* {

*if* (front == -1) {

            front = 0;

            rear = 0;

        } *else* *if* (rear == *MAX\_SIZE* - 1) {

            rear = 0;

        } *else* {

            rear = rear + 1;

        }

        deque[rear] = item;

*printf*("Inserted %d at the rear of the Deque.\n", item);

    }

}

int *deleteFront*() {

*if* (*isEmpty*()) {

*printf*("Deque is empty. Cannot delete from the front.\n");

*return* -1;

    } *else* {

        int data = deque[front];

*if* (front == rear) {

            front = -1;

            rear = -1;

        } *else* *if* (front == *MAX\_SIZE* - 1) {

            front = 0;

        } *else* {

            front = front + 1;

        }

*printf*("Deleted %d from the front of the Deque.\n", data);

*return* data;

    }

}

int *deleteLast*() {

*if* (*isEmpty*()) {

*printf*("Deque is empty. Cannot delete from the rear.\n");

*return* -1;

    } *else* {

        int data = deque[rear];

*if* (front == rear) {

            front = -1;

            rear = -1;

        } *else* *if* (rear == 0) {

            rear = *MAX\_SIZE* - 1;

        } *else* {

            rear = rear - 1;

        }

*printf*("Deleted %d from the rear of the Deque.\n", data);

*return* data;

    }

}

int *getFront*() {

*if* (*isEmpty*()) {

*printf*("Deque is empty. Cannot get front element.\n");

*return* -1;

    } *else* {

*return* deque[front];

    }

}

int *getRear*() {

*if* (*isEmpty*()) {

*printf*("Deque is empty. Cannot get rear element.\n");

*return* -1;

    } *else* {

*return* deque[rear];

    }

}

void *display*() {

*if* (*isEmpty*()) {

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

    } *else* {

*printf*("Deque elements: ");

        int i = front;

*do* {

*printf*("%d ", deque[i]);

            i = (i + 1) % *MAX\_SIZE*;

        } *while* (i != (rear + 1) % *MAX\_SIZE*);

*printf*("\n");

    }

}

int *main*() {

    int choice, item;

*while* (1) {

*printf*("\nChoose your Deque Operation:\n");

*printf*("1. Insert Front\n");

*printf*("2. Insert Rear\n");

*printf*("3. Delete Front\n");

*printf*("4. Delete Rear\n");

*printf*("5. Get Front\n");

*printf*("6. Get Rear\n");

*printf*("7. Display\n");

*printf*("8. Exit\n");

*printf*("Enter your choice: ");

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

*switch* (choice) {

*case* 1:

*printf*("Enter the element to be inserted at the front: ");

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

*insertFront*(item);

*break*;

*case* 2:

*printf*("Enter the element to be inserted at the rear: ");

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

*insertLast*(item);

*break*;

*case* 3:

*deleteFront*();

*break*;

*case* 4:

*deleteLast*();

*break*;

*case* 5:

*printf*("Front Element: %d\n", *getFront*());

*break*;

*case* 6:

*printf*("Rear Element: %d\n", *getRear*());

*break*;

*case* 7:

*display*();

*break*;

*case* 8:

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

*exit*(0);

*default*:

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

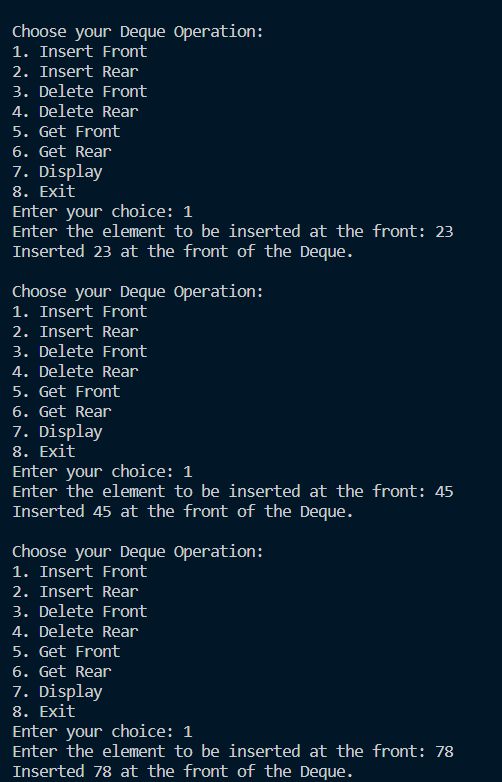
        }

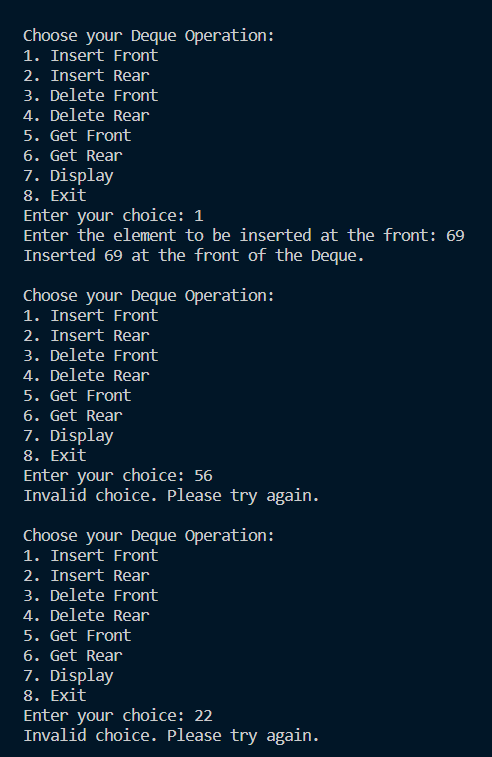
    }

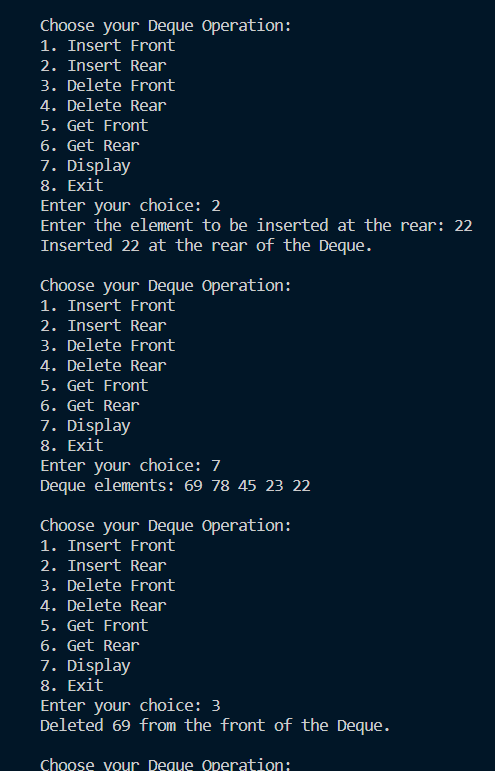
*return* 0;

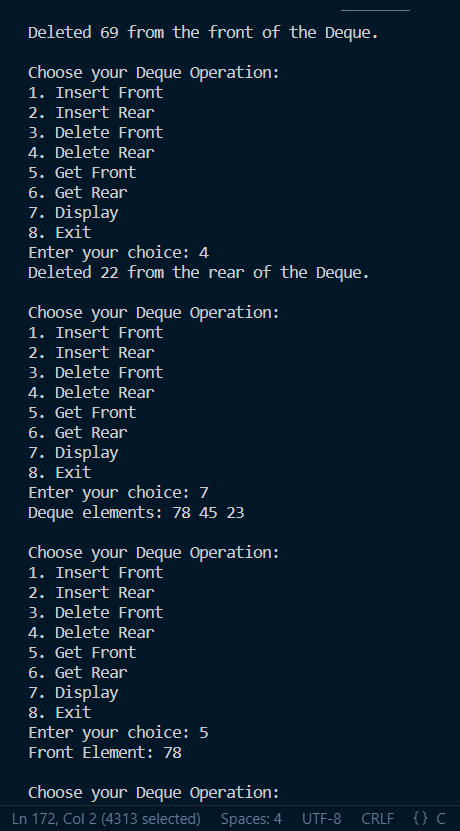
}

Output:











**Question 5:**

You are given a stack data structure with push and pop operations. Implement a queue using instances of stack data structure and operations on it.( ie, Implement queue using stack)

#*include* <stdio.h>

#*include* <stdlib.h>

#*define* *MAX* 10

int stack1[*MAX*], stack2[*MAX*];

int top1 = -1, top2 = -1;

void *enqueue*(int item) {

*if* (top1 == *MAX* - 1) {

*printf*("Queue is full\n");

    } *else* {

        stack1[++top1] = item;

    }

}

int *dequeue*() {

*if* (top1 == -1 && top2 == -1) {

*return* -1;// *Queue is empty*

    } *else* {

*if* (top2 == -1) {// *Transfer elements from stack1 to stack2*

*while* (top1 != -1) {

                stack2[++top2] = stack1[top1--];

            }

        }

        int item = stack2[top2--];// *Pop from stack2 (front of the queue)*

*return* item;

    }

}

void *display*() {

*if* (top1 == -1 && top2 == -1) {

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

    } *else* {

*printf*("The elements in the queue are:\n");

*for* (int i = 0; i <= top1; i++) {

*printf*("%d\n", stack1[i]);

        }

*for* (int i = top2; i >= 0; i--) {

*printf*("%d\n", stack2[i]);

        }

    }

}

int *main*() {

    int choice, item;

*while* (1) {

*printf*("\nChoose your queue "

               "Operation\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\n");

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

*switch* (choice) {

*case* 1:

*printf*("Enter the element to be inserted:");

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

*enqueue*(item);

*display*();

*break*;

*case* 2:

                item = *dequeue*();

*if* (item == -1) {

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

                } *else* {

*printf*("The deleted element is %d\n", item);

*display*();

                }

*break*;

*case* 3:

*display*();

*break*;

*case* 4:

*printf*("\nExit...");

*exit*(0);

*default*:

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

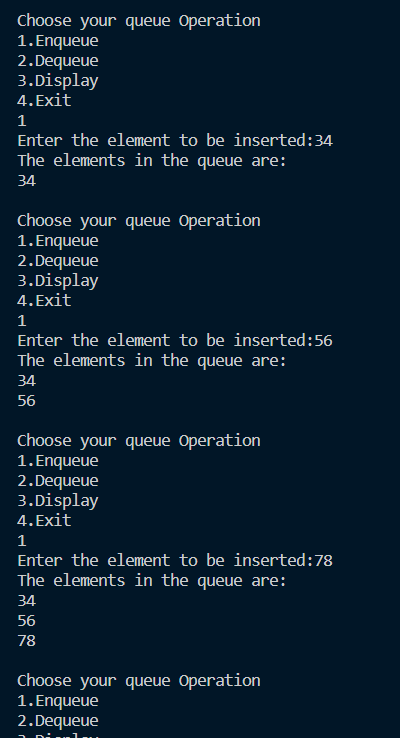
        }

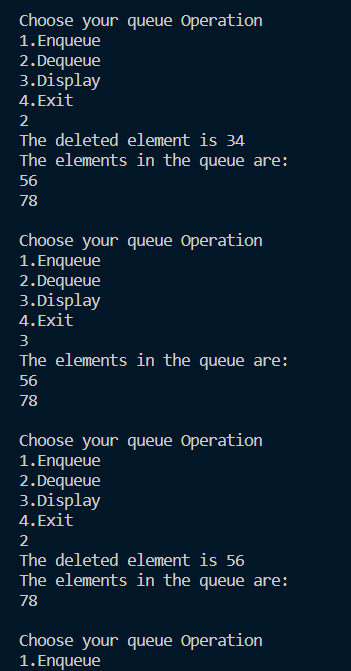
    }

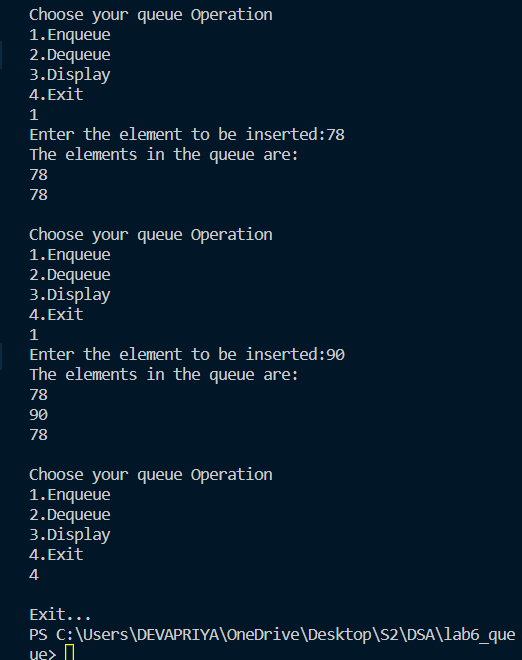
*return* 0;

}

Output:





****

===========================================