Name: - Atish Kumar

Roll No: - 120CS0173

Lab Sheet: - 07

Q1. Write a program for implementation of priority queue with character data elements. Each element has an integer priority number between 1-5. Implement Enqueue () and Dequeue () functions in each of the following cases.

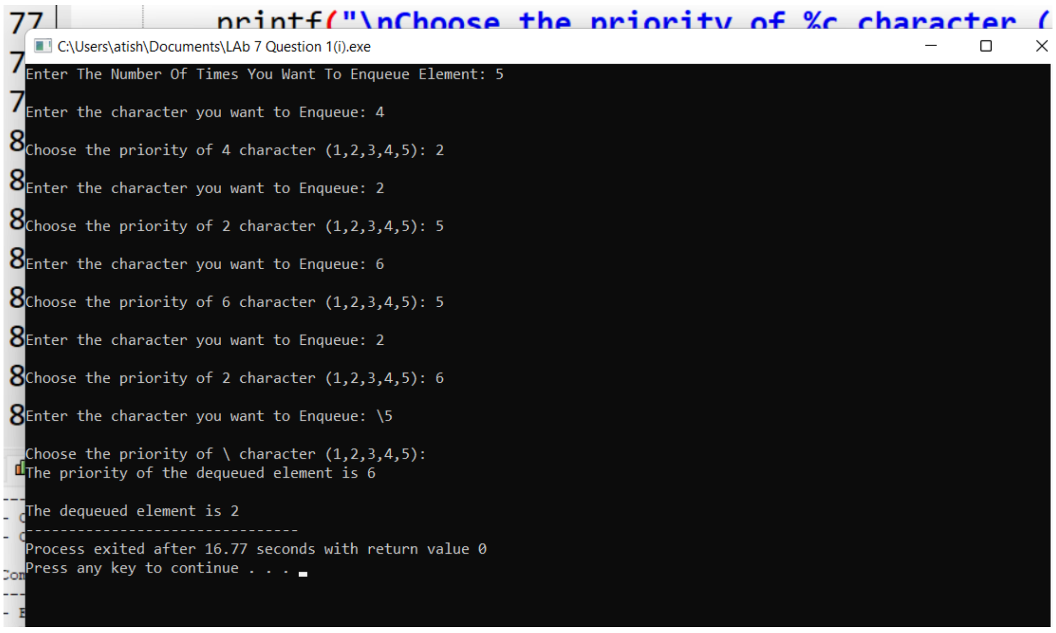
1. Linked list implementation of priority queue

Program:-

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

struct Queue{
    char character;
    int priority;
    struct Queue *next;
};

struct Queue *Enqueue(struct Queue *head, char ch, int p)
{
    struct Queue *h = head;
    struct Queue *q = (struct Queue *)malloc(sizeof(struct Queue));
    if(q == NULL)
    printf("The queue is full and cannot insert more element\n");
    else
    {
            q->character = ch;
            q->priority = p;
            if(h == NULL)
            {
                h = q;
                h->next = NULL;
                return h;
            }
            else if (p > h->priority)
            {
                q->next = h;
                h = q;
                return h;
            }
            else
            {
                while (h->next != NULL && h->next->priority >= p)
                {
                    h = h->next;
                }
                q->next = h->next;
                h->next = q;
                return head;
            }
        }
}

char Dequeue(struct Queue **head)
```

```
{
    char ch;
    struct Queue *q = *head;
    if (q == NULL)
    {
            printf("The queue is empty and cannot be dequeued\n");
            return ch;
    }
    else
    {
            ch = q->character;
            printf("\nThe priority of the dequeued element is %d\n", q-
>priority);
            *head = (*head)->next;
            free(q);
    }
    return ch;
}

int main()
{
    struct Queue *head = NULL;
    int n,i,d, p;
    char c, ch;
    printf("Enter The Number Of Times You Want To Enqueue Element: ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
            printf("\nEnter the character you want to Enqueue: ");
            getchar();
            scanf("%c", &ch);
            printf("\nChoose the priority of %c character (1,2,3,4,5): ", ch);
            scanf("%d", &p);
            head = Enqueue(head, ch, p);
    }
    for(i=0; i<d; i++)
    {
            c = Dequeue(&head);
            printf("\nThe dequeued element is %c", c);
    }
    return 0;
}
```

Output:-



```
C:\Users\atish\Documents\LAb 7 Question 1(i).exe
Enter The Number Of Times You Want To Enqueue Element: 5

Enter the character you want to Enqueue: 4

Choose the priority of 4 character (1,2,3,4,5): 2

Enter the character you want to Enqueue: 2

Choose the priority of 2 character (1,2,3,4,5): 5

Enter the character you want to Enqueue: 6

Choose the priority of 6 character (1,2,3,4,5): 5

Enter the character you want to Enqueue: 2

Choose the priority of 2 character (1,2,3,4,5): 6

Enter the character you want to Enqueue: \5

Choose the priority of \ character (1,2,3,4,5):
The priority of the dequeued element is 6

The dequeued element is 2
--------------------------------
Process exited after 16.77 seconds with return value 0
Press any key to continue . . .
```

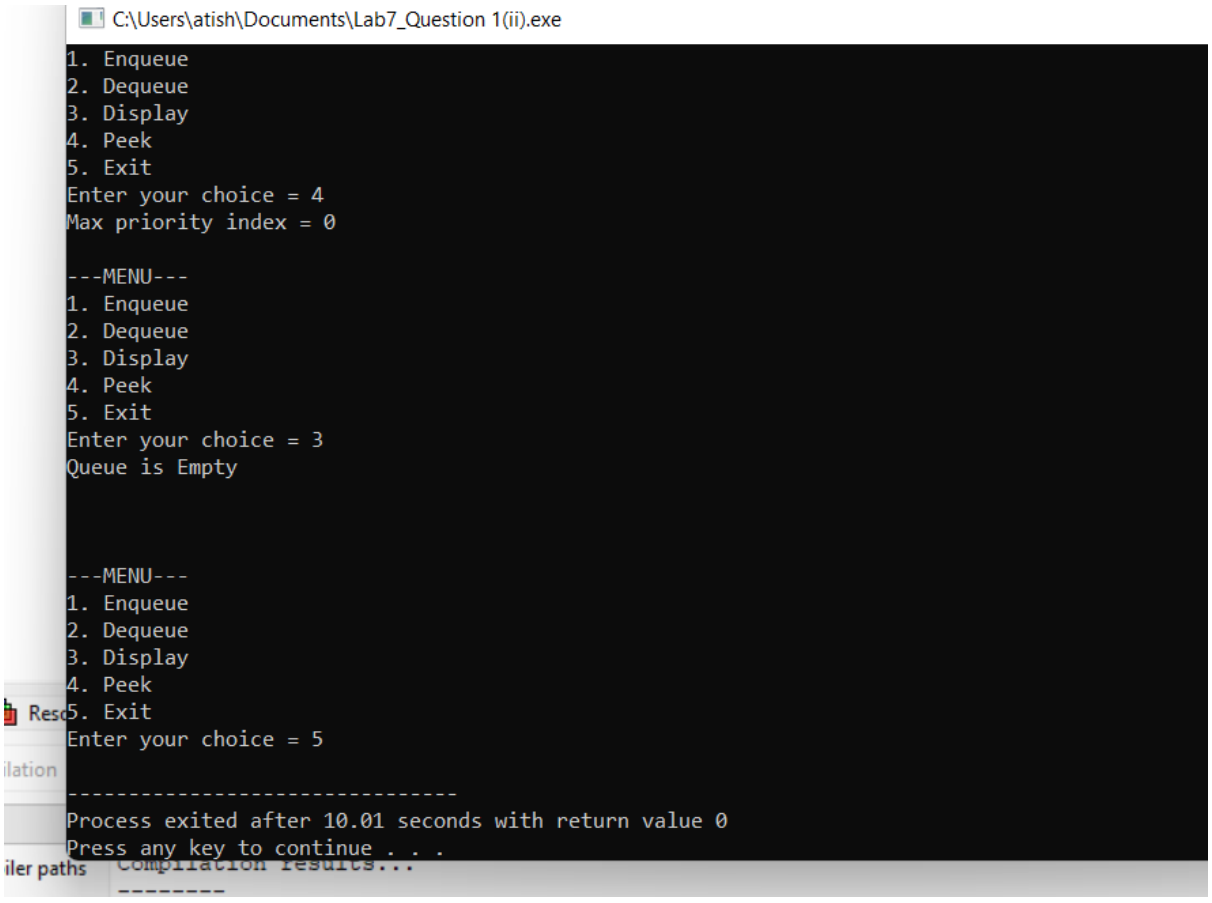
2. Array implementation of priority queue.

Program:-

```
#include<stdio.h>
#include <stdlib.h>
#define SIZE 5

struct item
{
  char data;
  int priority;
};

struct item pr_queue[SIZE];
int front = -1;
int rear = -1;

void enqueue()
{
	char data;
  int priority;
  if(rear==SIZE-1)
  {
		printf("OVERFLOW!!\n\n");
  }
  else
  {
		printf("Enter a char and its priority:\n");
		scanf(" %c %d", &data, &priority);
		rear++;

		pr_queue[rear].data=data;
		pr_queue[rear].priority=priority;
		printf("\n\n");
  }
}

int peek()
{
	int max=0;
  int i;
  for(i=front;i<=rear;i++)
  {
                 if(pr_queue[max].priority<pr_queue[i].priority)
    {
                          max=i;
    }
  }
  return max;
}

void display()
{
  int i;
  if(rear == -1)
  {
                 printf("Queue is Empty\n\n");
  }
  else
  {
    for(i=front;i<=rear;i++)
    {
                          printf(" %c", pr_queue[i].data);
    }
  }
         printf("\n\n");
}

void dequeue()
{
  int index=peek();
  int i;
  if(rear == -1)
  {
    printf("UNDERFLOW!!\n\n");
  }
  else
  {

    for(i=index; i<=rear; i++)
    {
                          pr_queue[i]=pr_queue[i+1];
    }
    rear--;
  }
  printf("\n\n");
```

```
}

int main()
{
	front++;
	int ch;
	char data;
	int priority;
	while (1)
	{
		printf("---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", &ch);
		switch(ch)
		{
			case 1:
				enqueue();
				break;
			case 2:
				dequeue();
				break;
			case 3:
				display();
				break;
			case 4:
				printf("Max priority index = %d\n\n", peek());
				break;
			case 5:
				exit(0);
			default:
				printf("Wrong choice!!\n");
		}
	}
	return 0;
}
```

Output:-



C:\Users\atish\Documents\Lab7_Question 1(ii).exe

```
1. Enqueue
2. Dequeue
3. Display
4. Peek
5. Exit
Enter your choice = 4
Max priority index = 0

---MENU---
1. Enqueue
2. Dequeue
3. Display
4. Peek
5. Exit
Enter your choice = 3
Queue is Empty



---MENU---
1. Enqueue
2. Dequeue
3. Display
4. Peek
5. Exit
Enter your choice = 5

--------------------------------
Process exited after 10.01 seconds with return value 0
Press any key to continue . . .
```

Q2. Write a program to implement circular queue using Arrays.

Program:-

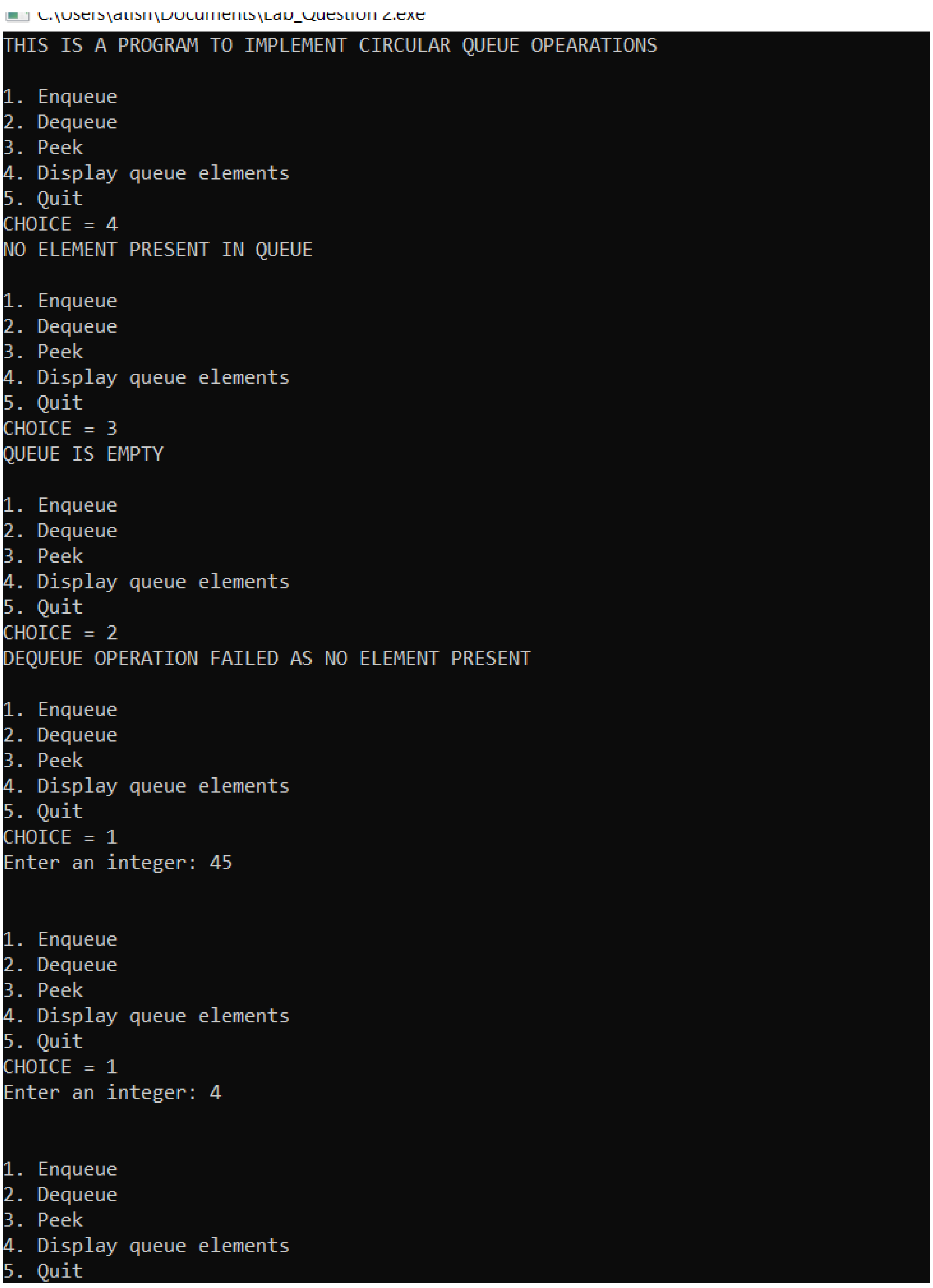
```
#include <stdio.h>
#include <stdlib.h>

void enqueue();
void dequeue();
void peek();
void display();

short front = -1, rear = -1;
int cque[5];

int main()
{
	char choice;
	printf("THIS IS A PROGRAM TO IMPLEMENT CIRCULAR QUEUE
OPEARATIONS\n\n");
	while(1)
	{
		printf("1. Enqueue\n2. Dequeue\n3. Peek\n4. Display queue
elements\n5. Quit\n");
		printf("CHOICE = ");
		scanf(" %c", &choice);
		switch(choice)
		{
			case '1':
				enqueue();
				printf("\n\n");
				break;
			case '2':
				dequeue();
				printf("\n\n");
				break;
			case '3':
				peek();
				printf("\n\n");
				break;
			case '4':
				display();
				printf("\n\n");
				break;
```

```
                    case '5':
                            exit(0);
                            break;
                    default:
                            printf("INVALID CHOICE\n\n");
            }
        }
        return 0;
}

void enqueue()
{
        int ele;
        if(front==-1 && rear==-1)
        {
                ++front;
                ++rear;
                printf("Enter an integer: ");
                scanf("%d", &ele);
                cque[rear] = ele;
        }
        else if(front==0 && rear>=0 && rear<4)
        {
                ++rear;
                printf("Enter an integer: ");
                scanf("%d", &ele);
                cque[rear] = ele;
        }
        else if((front==0 && rear==4) || (front==(rear+1)))
        printf("ENQUEUE OPERATION FAILED AS QUEUE IS FULL");
        else if(front>0 && rear<4 && front<=rear)
        {
                ++rear;
                printf("Enter an integer: ");
                scanf("%d", &ele);
                cque[rear] = ele;
        }
        else if(front>0 && front<=4 && rear==4)
        {
                rear = 0;
                printf("Enter an integer: ");
                scanf("%d", &ele);
                cque[rear] = ele;
        }
```

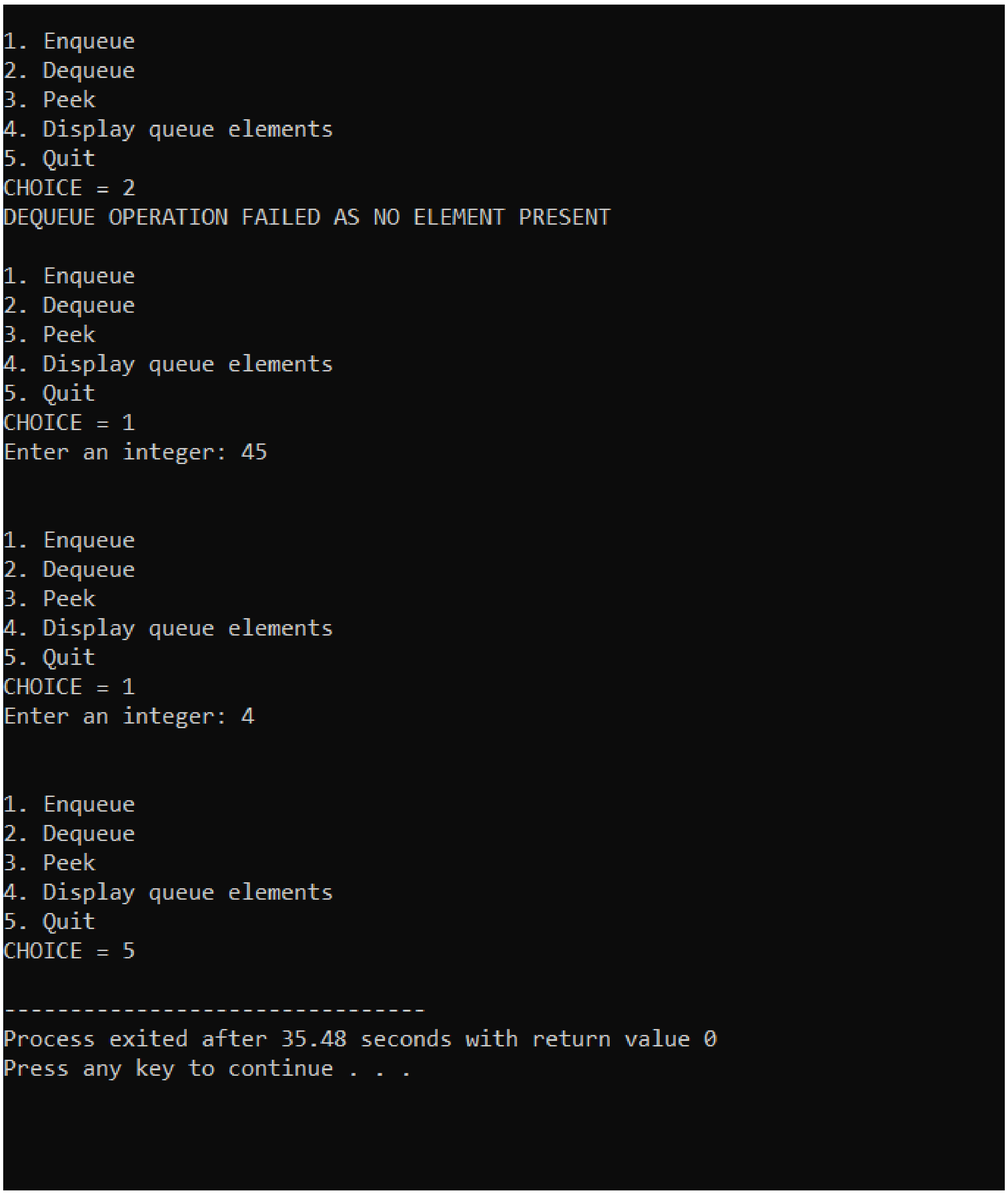
```
	else if(rear>=0 && front>1 && front<=4)
	{
		++rear;
		printf("Enter an integer: ");
		scanf("%d", &ele);
		cque[rear] = ele;
	}
}

void dequeue()
{
	if(front==-1 && rear==-1)
	printf("DEQUEUE OPERATION FAILED AS NO ELEMENT PRESENT");
	else if(front>=0 && rear<=4 && front<rear)
	printf("Dequeued element = %d", cque[front++]);
	else if((front==rear) && (front!=-1 && rear!=-1))
	{
		printf("Dequeued element = %d", cque[front]);
		front = -1;
		rear = -1;
	}
	else if(rear>0 && rear<=4 && front==4)
	{
		printf("Dequeued element = %d", cque[front]);
		front = 0;
	}
	else if(rear>=0 && front<4 && rear<front)
	printf("Dequeued element = %d", cque[front++]);
}

void peek()
{
	if(front==-1 && rear==-1)
	printf("QUEUE IS EMPTY");
	else if(front!=rear)
	{
		printf("First element in queue = %d\n", cque[front]);
		printf("Last element in queue = %d", cque[rear]);
	}
	else
	printf("First and last element of the queue = %d", cque[front]);
}

void display()
```

```
{
        short i;
        if(front==-1 && rear==-1)
        printf("NO ELEMENT PRESENT IN QUEUE");
        else if(front>=0 && rear<=4 && front<rear)
        {
                printf("Queue elements are:\n");
                for(i=front; i<=rear; i++)
                {
                        printf("%d\t", cque[i]);
                }
        }
        else if(rear>=0 && front<=4 && rear<front)
        {
                printf("Queue elements are:\n");
                for(i=front; i<=4; i++)
                {
                        printf("%d\t", cque[i]);
                }
                for(i=0; i<=rear; i++)
                {
                        printf("%d\t", cque[i]);
                }
        }
        else if((front==rear) && (front!=-1 && rear!=-1))
        printf("Only element in queue = %d", cque[front]);
}
```

OutPut;-



```
THIS IS A PROGRAM TO IMPLEMENT CIRCULAR QUEUE OPEARATIONS

1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 4
NO ELEMENT PRESENT IN QUEUE

1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 3
QUEUE IS EMPTY

1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 2
DEQUEUE OPERATION FAILED AS NO ELEMENT PRESENT

1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 1
Enter an integer: 45


1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 1
Enter an integer: 4


1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
```



```
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 2
DEQUEUE OPERATION FAILED AS NO ELEMENT PRESENT

1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 1
Enter an integer: 45


1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 1
Enter an integer: 4


1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit
CHOICE = 5

--------------------------------
Process exited after 35.48 seconds with return value 0
Press any key to continue . . .
```

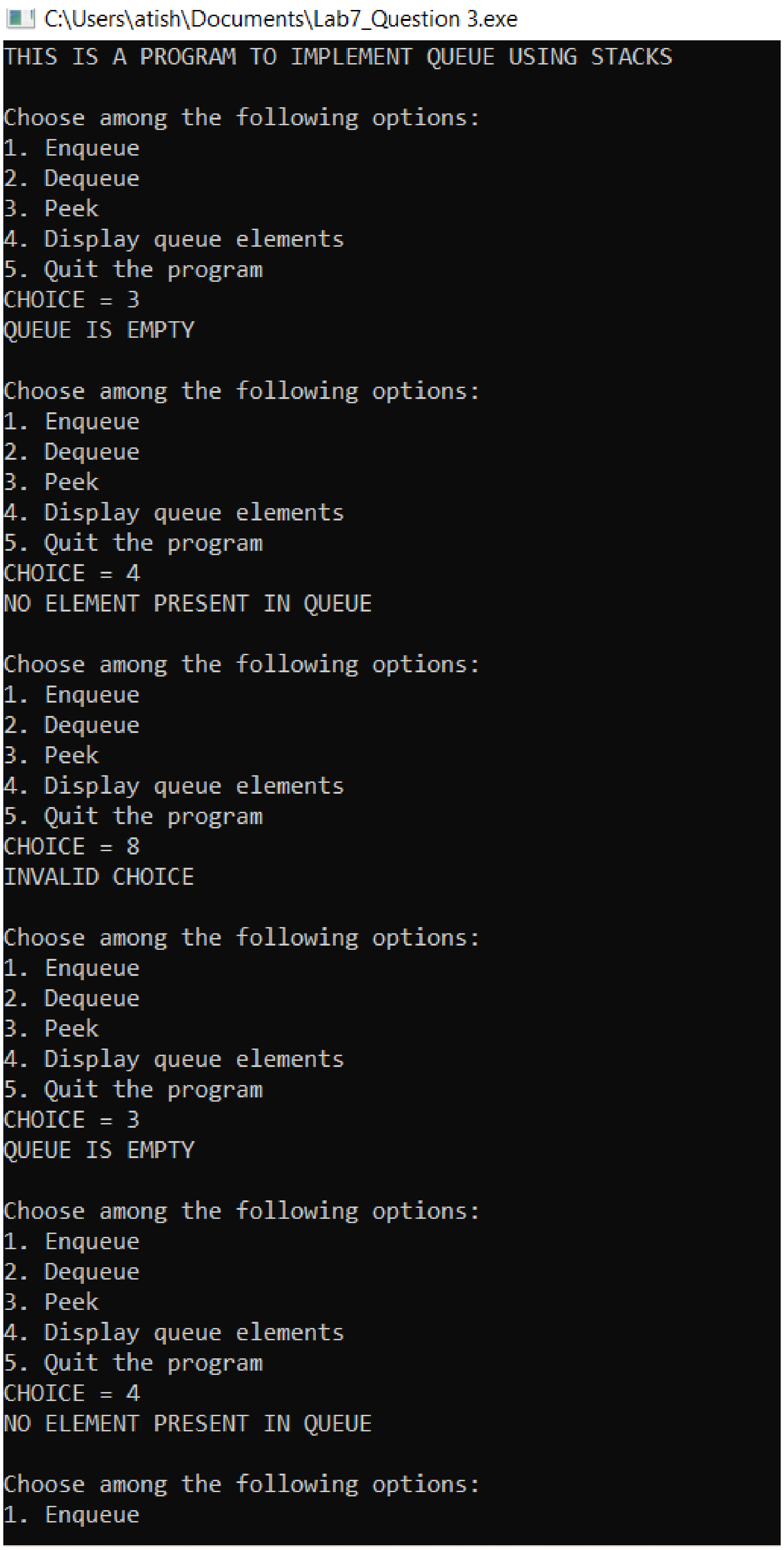
Q3 Write a program to implementation a Queue using stacks.
Program:-
```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4

void enqueue();
void dequeue();
void peek();
void display();

short i, j, top1=-1, top2=-1;
int stack1[SIZE], stack2[SIZE];

int main()
{
	char choice;
	printf("THIS IS A PROGRAM TO IMPLEMENT QUEUE USING STACKS\n\n");

	/*IN THIS PROGRAM, OPERATIONS ON STACK SUCH AS push(), pop() and
peek()
	  ARE MODIFIED AND CLEVERLY HIDDEN WITHIN THE QUEUE OPERATIONS
enqueue()
	  and dequeue().*/

	while(1)
	{
		printf("Choose among the following options:\n");
		printf("1. Enqueue\n2. Dequeue\n3. Peek\n4. Display queue
elements\n5. Quit the program\nCHOICE = ");
		scanf(" %c", &choice);
		switch(choice)
		{
			case '1':
				enqueue();
				printf("\n\n");
				break;
			case '2':
				dequeue();
				printf("\n\n");
				break;
			case '3':
				peek();
```

```
					printf("\n\n");
					break;
				case '4':
					display();
					printf("\n\n");
					break;
				case '5':
					exit(0);
					break;
				default:
					printf("INVALID CHOICE\n\n");
			}
		}
		return 0;
}

void enqueue()
{
		int ele;
		if(top1>=-1 && top1<SIZE-1)
		{
			++top1;
			printf("Enter an integer: ");
			scanf("%d", &ele);
			stack1[top1] = ele;
		}
		else if(top1==SIZE-1	&& top2==-1)
		{
			j = top1;
			for(i=0; i<=top1; i++)
			{
				stack2[j-i] = stack1[i];
				top2++;
			}
			top1 = 0;
			printf("Enter an integer: ");
			scanf("%d", &ele);
			stack1[top1] = ele;
		}
		else if(top1==SIZE-1 && top2>=0 && top2<SIZE-1)
		{
			j = SIZE-top2-1;
			for(i=0; i<j; i++)
			{
				stack2[i+j] = stack2[i];
			}
			top2 += j;
			for(i=0; i<j; i++)
			{
				stack2[j-i-1] = stack1[i];
			}
			for(i=0; i<j; i++)
			{
				stack1[i] = stack1[i+j];
			}
			top1 -= j;
			printf("Enter an integer: ");
			scanf("%d", &ele);
			stack1[++top1] = ele;
		}
		else if(top1==SIZE-1 && top2==SIZE-1)
		printf("ENQUEUE OPERATION FAILED AS QUEUE IS FULL");
}

void dequeue()
{
		if(top1==-1 && top2==-1)
		printf("DEQUEUE OPERATION FAILED AS QUEUE IS EMPTY");
		else if(top1!=-1 && top2==-1)
		{
			j = top1;
			for(i=0; i<=top1; i++)
			{
				stack2[j-i] = stack1[i];
				top2++;
			}
			top1 = -1;
			printf("Dequeued element = %d", stack2[top2--]);
		}
		else if(top1==-1 && top2!=-1)
		{
			printf("Dequeued element = %d", stack2[top2--]);
		}
		else if(top1!=-1 && top2!=-1)
		{
			printf("Dequeued element = %d", stack2[top2--]);
		}
```

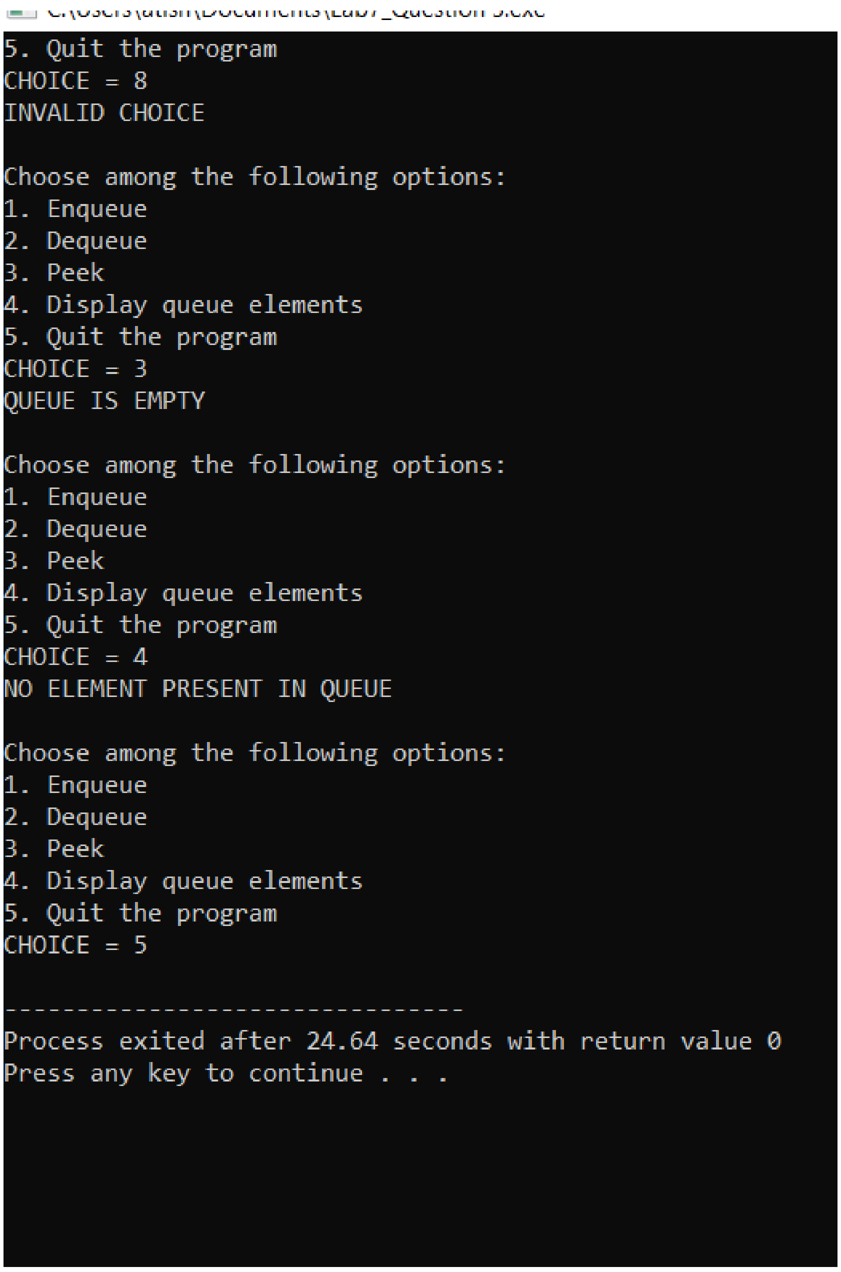
```
}

void peek()
{
        if(top1==-1 && top2==-1)
        printf("QUEUE IS EMPTY");
        else if(top1!=-1 && top2==-1)
        {
                printf("First element in queue = %d\n", stack1[0]);
                printf("Last element in queue = %d", stack1[top1]);
        }
        else if(top1==-1 && top2!=-1)
        {
                printf("First element in queue = %d\n", stack2[top2]);
                printf("Last element in queue = %d", stack2[0]);
        }
        else if(top1!=-1 && top2!=-1)
        {
                printf("First element in queue = %d\n", stack2[top2]);
                printf("Last element in queue = %d", stack1[top1]);
        }
}

void display()
{
        if(top1==-1 && top2==-1)
        printf("NO ELEMENT PRESENT IN QUEUE");
        else if(top1!=-1 && top2==-1)
        {
                printf("Queue elements are:\n");
                for(i=0; i<=top1; i++)
                {
                        printf("%d\t", stack1[i]);
                }
        }
        else if(top1==-1 && top2!=-1)
        {
                printf("Queue elements are:\n");
                for(i=top2; i>=0; i--)
                {
                        printf("%d\t", stack2[i]);
                }
        }
        else if(top1!=-1 && top2!=-1)
```

```
	{
		printf("Queue elements are:\n");
		for(i=top2; i>=0; i--)
		{
			printf("%d\t", stack2[i]);
		}
		for(j=0; j<=top1; j++)
		{
			printf("%d\t", stack1[j]);
		}
	}
}
```

Output:-



```
C:\Users\atish\Documents\Lab7_Question 3.exe
THIS IS A PROGRAM TO IMPLEMENT QUEUE USING STACKS

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 3
QUEUE IS EMPTY

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 4
NO ELEMENT PRESENT IN QUEUE

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 8
INVALID CHOICE

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 3
QUEUE IS EMPTY

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 4
NO ELEMENT PRESENT IN QUEUE

Choose among the following options:
1. Enqueue
```


C:\Users\atish\Documents\Lab7_Question 3.exe

```
5. Quit the program
CHOICE = 8
INVALID CHOICE

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 3
QUEUE IS EMPTY

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 4
NO ELEMENT PRESENT IN QUEUE

Choose among the following options:
1. Enqueue
2. Dequeue
3. Peek
4. Display queue elements
5. Quit the program
CHOICE = 5

--------------------------------
Process exited after 24.64 seconds with return value 0
Press any key to continue . . .
```