**Name: KAMCHE YANN ARNAUD**

**Matricule : FE21A208**

**Department: Computer Engineering**

**Level: 300**

**Task: Implement a queue using array**

1. **CODE**

/\* IMPLEMENTATION OF QUEUE

Using Arrays

10/20/2022

\*/

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#define MAX\_SIZE 100

int A[MAX\_SIZE];

int front = -1;

int rear = -1;

int result;

//Create function

void Create(int Queue\_Size)

{

int A[Queue\_Size];

return;

}

/\*Enqueue inserts an element into the queue

Elements are inserted from the rear.

The are three conditions to check in this case, which are:

- Is the queue full, if yes display the state of the queue being 'FULL'

- Is there only one element, if yes, both the front and the rear will automatically

acquire the index of the first position in the queue.

- Are there more than one element, if yes, increment the rear and insert an element

\*/

//Queue\_Full: Queue is completely full

void Queue\_Full()

{

printf("Error: QUEUE OVERFLOW\n");

return;

}

//Queue\_Empty: Nothing in the queue

void Queue\_Empty()

{

printf("Error: QUEUE EMPTY\n");

return;

}

void Enqueue(int max\_size, int element)

{

if (rear == MAX\_SIZE-1)

Queue\_Full();

else if (front == -1 && rear == -1)

rear = front = 0;

else

++rear;

A[rear] = element;

return;

}

/\*Dequeue removes or pops an element out of the queue

Elements are removed from the front

There are three conditions to check in this case, which are:

- Is the queue empty, if yes, display the state of the queue being 'EMPTY'

- Is there only one element in the queue, if yes, both the rear and the front

will acquire the null index(-1), signifying that the queue is empty

- Are there more than one element in the queue, increment the front.

\*/

int Dequeue()

{

if(front == -1)

return 0;

else

return A[front++];

}

//Display: Prints the content of the queue

void Display()

{

int i;

if( front == -1 )

Queue\_Empty();

else

{

printf("Queue: ");

for(i = front; i <= rear ; i++)

printf("%d ", A[i]);

printf("\n");

}

return;

}

//headOfQueue: Displays the element at the front of the queue

int headOfQueue()

{

if(front == -1)

return 0;

else

return A[front];

}

//endOfQueue: Displays the element at the rear of the queue

int endOfQueue()

{

if(rear == -1)

return 0;

else

return A[rear];

}

//sizeOfQueue: Displays the size of the Queue

int sizeOfQueue()

{

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

return 0;

else

return rear - front +1;

}

//Status\_Queue: Displays the state of the queue

void statusOfQueue(int size)

{

if(front == -1)

Queue\_Empty();

else if (rear == size - 1)

Queue\_Full();

else

printf("STATUS: Not Empty");

return;

}

int main()

{

int Queue\_Size, choice, num;

system("color 2");

printf("Enter the size of your Queue: ");

scanf("%d", &Queue\_Size);

Create(Queue\_Size);

printf("1. Enqueue\n");

printf("2. Dequeue\n ");

printf("3. Head of queue\n");

printf("4. End of queue\n");

printf("5. Display Queue\n");

printf("6. Size of queue\n");

printf("7. Status of queue\n");

options:

printf("\nChoose the operation to be performed with your list: ");

scanf("%d", &choice);

while(choice == 1|| choice == 2|| choice == 3|| choice == 4 || choice == 5 || choice == 6 || choice == 7)

{

while(choice==1)

{

printf("Enter a number: ");

scanf("%d", &num);

Enqueue(Queue\_Size, num);

goto options;

}

while(choice == 2){

result = Dequeue();

if(result == 0 )

printf("Queue is empty");

else

printf("%d", result);

goto options;

}

while(choice == 3){

result = headOfQueue();

if(result == 0)

printf("No Element at the front");

else

printf("%d", result);

goto options;

}

while(choice == 4){

result = endOfQueue();

if(result == 0)

printf("No Element at the end");

else

printf("%d", result);

goto options;

}

while(choice == 5){

Display();

goto options;

}

while(choice == 6){

result= sizeOfQueue();

if(result == 0)

printf("Queue is empty");

else

printf("%d", result);

goto options;

}

while(choice == 7){

statusOfQueue(Queue\_Size);

goto options;

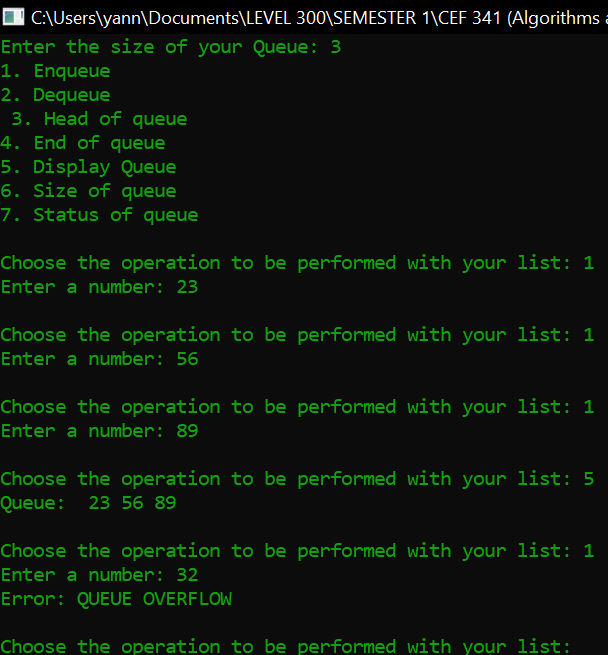
}

}

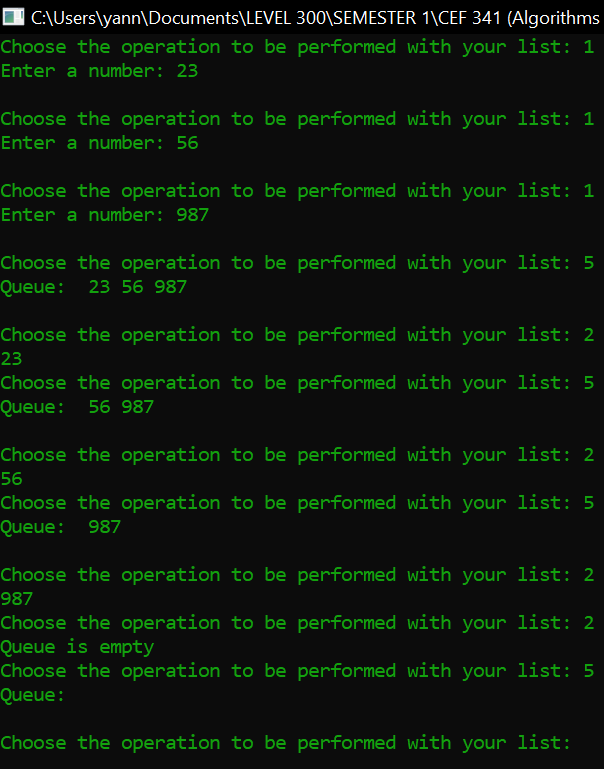
return 0;

}

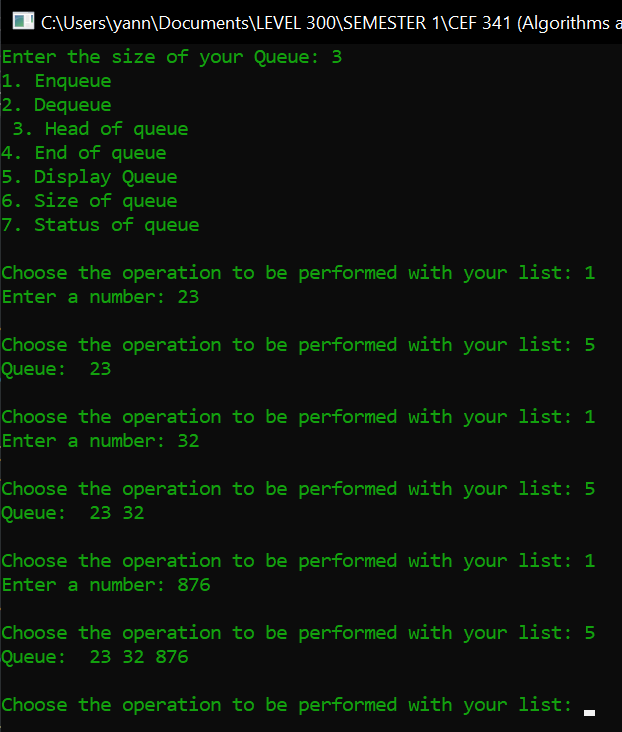
1. **COMPILATION RESULTS**
2. **ENQUEUE OPERATION**

****

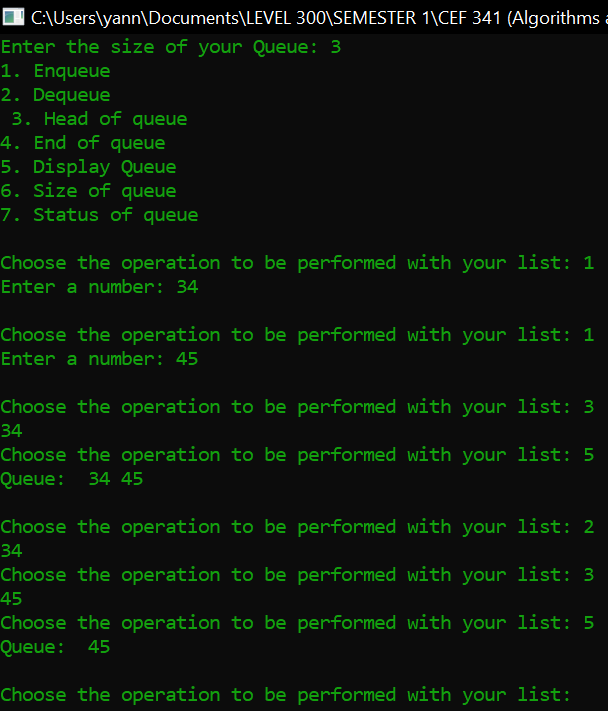
1. **DEQUEUE OPERATION**

****

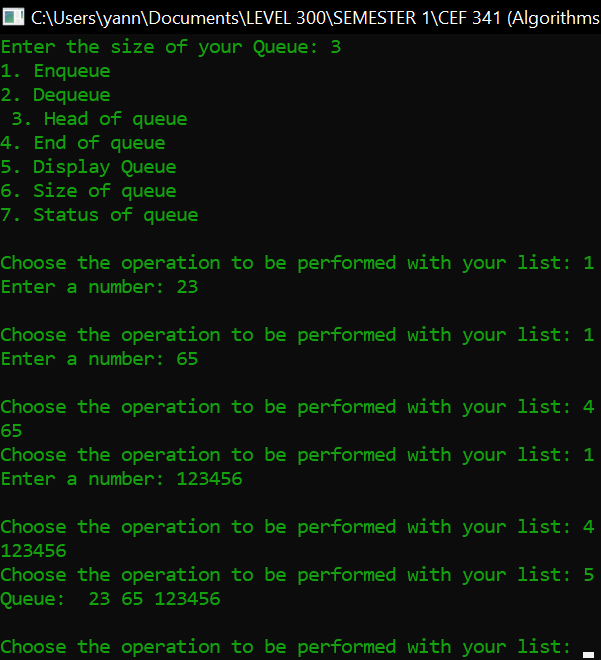
1. **DISPLAY QUEUE**

****

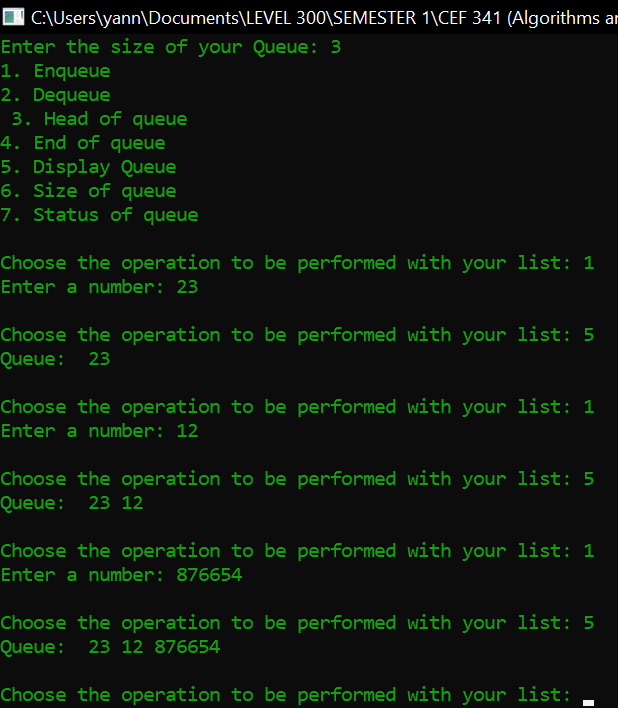
1. **HEAD OF QUEUE**

****

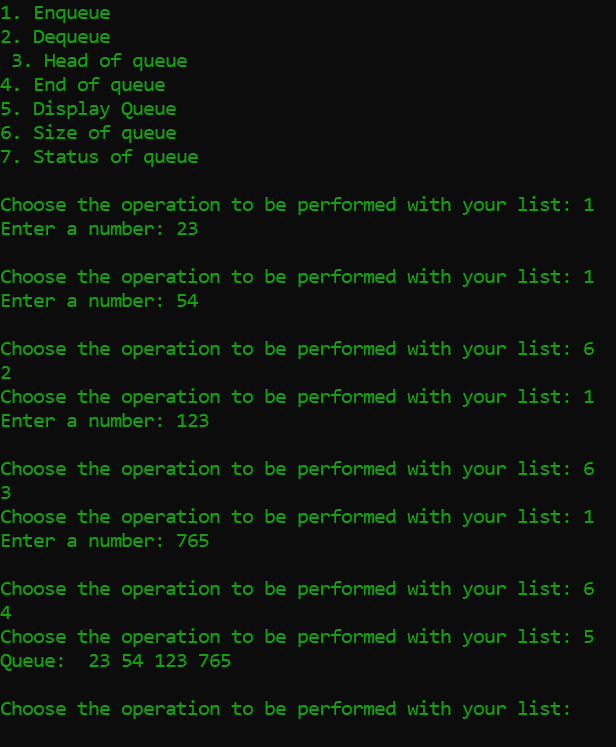
1. **END OF QUEUE**

****

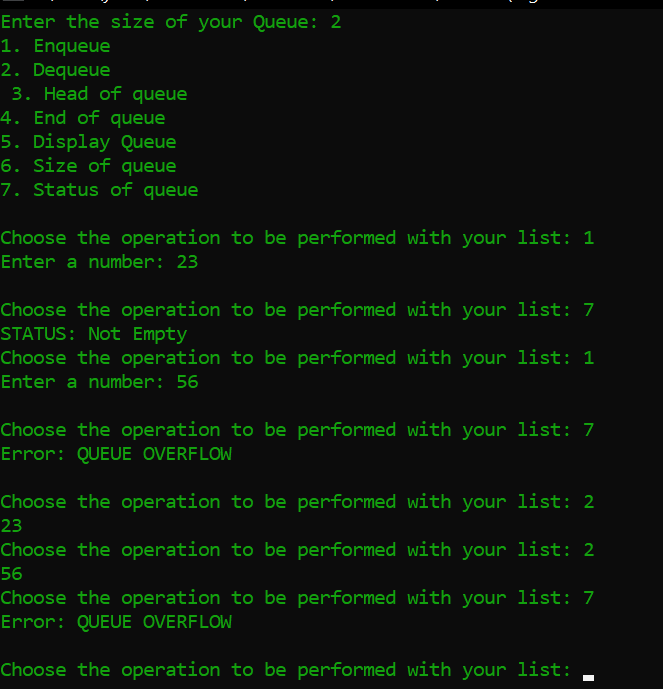
1. **DISPLAY QUEUE**

****

1. **SIZE OF QUEUE**

****

1. **STATUS OF QUEUE**

****