**CHAPTER 1**

# INTRODUCTION

## 1.1 Scope and Importance

Computer graphics deals with generating images with the aid of computers. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer displays, and many specialized applications. It is a vast and recently developed area of computer science. Computer graphics is responsible for displaying art and image data effectively and meaningfully to the consumer. It is also used for processing image data received from the physical world, such as photo and video content. Computer graphics development has had a significant impact on many types of media and has revolutionized animation, movies, advertising, video games, in general.

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn). It gives software developers access to geometric and image primitives, display lists, modelling transformations, lighting and texturing, anti-aliasing, blending, and many other features. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

accomplish our project in an effective manner. It’s a simple computer graphics project which is useful to understand the stack and queue data structures easily.

## 1.2 Problem Definition

In this Project, we are writing a program in C++ and OpenGL to represent Stack and Queue and show all its operations. Once user makes a choice between the options i.e Stack and Queue, He can enter the desired size. In stack we can perform four different operations which include; PUSH, POP, IsEmpty and IsFull. Whereas the four operation performed on queue includes; ENQUEUE (insert), DEQUEUE (delete), Overflow and Underflow. Once the user is done performing all the function he can exit using EXIT button.

## 1.3 Hardware Requirement Specification

The interface for the 2D package requires for the user to have a mouse connected, and the corresponding drivers software and header files installed. Mouse and Keyboard is taken as the input device for the user.

* PROCESSOR: Intel® Core™ i5-1035G1 CPU@ 1.00GHz 1.19
* RAM: 2.00 GB Minimum
* SPACE ON DISC: 4GB Minimum
* Keyboard  Mouse

## 1.4 Software Requirement Specification

* OpenGL libraries
* Operating System: Windows, Linux, macOS
* Languages: C++ using OpenGL  Code Blocks

**CHAPTER 2**

### ABOUT THE PROJRCT TITILE

### 2.1 Overview of Project

This project uses OpenGl and C++ to visualize the working of stack and queue data structure. We are implementing it using different OpenGL primtives, OpenGL libraries and combining them in a required manner. Stack and Queue are linear data structures. Stack follows LIFO (Last In First Out) property, where Last element to be inserted will be first to be deleted. On the other hand Queue follows FIFO (First In First Out) property, where first element to be inserted will be deleted first. In this project we will illustrate the different operations performed by these two data structures**.** It illustrates the role of different callback functions that provides easier way to

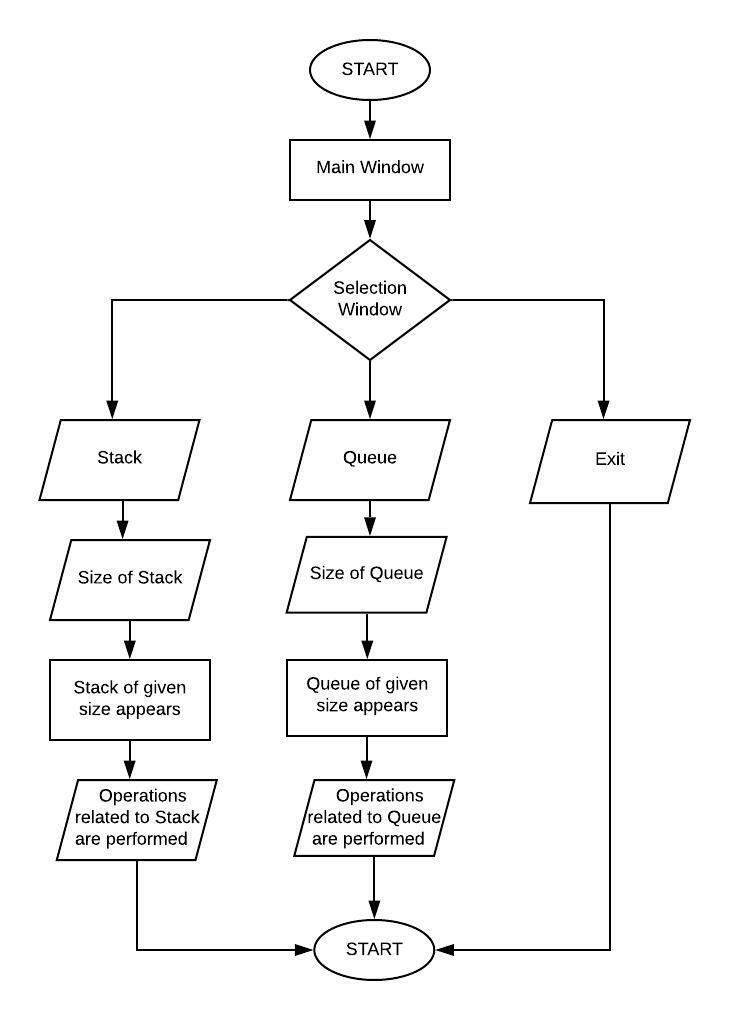
accomplish our project in an effective manner. It’s a simple computer graphics project which is useful to understand the stack and queue data structures easily.

## 2.2 Functional Requirements

For the execution of this project, the graphics have been written in C++ language and many simple user defined functions are used. This header file, in addition to the usual header files is needed for the working of the project. For running the project any basic C++ running compatible version of Code Blocks or Linux (Ubuntu) based platform is sufficient. We also added keyboard and mouse callback function for taking inputs from the user.

# 2.3 SYSTEM DESIGN

**Flow Chart**



**Figure 2.3:** Flow Chart for Stack and Queue Implementation

The above flowchart describes the architecture of this project; it helps us understand the steps involved and the features that have been implemented. It gives a clear idea of how user can interact and the choices that are provided to the user. The flowchart shows the destination the user arrives depending on his choices. Here the choices given are Stack, Queue and Exit and user can perform the operations that are related to his choices. Finally user can exit once he is done performing all the functions. Main components of the flow chart are explained below

## 2.3.1 Main Window

glutInit is used to initialize GLUT library and negotiate a session with the window system. glutInitWindowPosition and glutInitWindowSizeset gives the window position and size respectively. Windows created by glutCreateWindow will be requested to be created with the current initial window position and size. Main window will have the title of the project along with students name and USN. It will also contain a PROCEED button, when user clicks on it he will be moved to next window containing option including Stack, Queue or Exit. We use mouse callback function and keyboard callback function to take the input from the user.

## 2.3.2 Stack

Stack will be initialized using an array, we are going to implement different operation, and it contains only one pointer called TOP, that, points to the last or topmost element of the stack. Project shows the visual of how TOP moves when an element is inserted or delete. Once the user selects the size of the stack he can perform following operation.

Operations performed in Stack are as follows:

* **PUSH operation:** initially TOP will be pointing to the index 0 of the array, as the user inserts or pushes element into the stack then TOP will be incremented. When the TOP becomes equal to MAX then we cannot insert anymore elements.
* **POP operation:** User can delete elements from the stack using POP operation. When user enters the element that he wants to delete, TOP of the stack will be decremented and the selected element will be deleted.
* **OVERFLOW operation:** As user enters the elements to the stack and the TOP pointer becomes equal to MAX then the stack is said to be full. When this occurs we get a stack overflow message and user cannot enter anymore elements.
* **UNDERFLOW operation:** Once user deletes all the elements present then TOP pointer becomes equal to index 0. When this occurs user will get a Stack underflow message indicating stack is empty hence cannot delete anymore elements.

## 2.3.3 Queue

Queue is initialized with an array, it has 2 pointers that are FRONT and REAR, and here elements are added from the REAR end of the queue and deleted from the FRONT end. Unlike stack in queue we use both ends to perform different operation. Once the user selects the queue size he can perform different operation. Project shows the movement of both FRONT and REAR pointer as the elements are inserted and deleted from queue. Graphical representation of following operation will be shown in this project.

Operations performed in Queue are as follows:

* **ENQUEUE operation:** when user enters an element into queue first the algorithm checks if the queue is full, if not then the REAR pointer will be incremented by 1 and the element will be inserted into that index. As more elements are inserted we can see the movement of rear end and data is entered into next available space.
* **DEQUEUE operation:** when user wants to delete element present in queue, the algorithm will check if queue is empty, if not then the FRONT pointer will be decremented by 1 and the desired element will be deleted. As the elements are deleted from queue we can see the movement of Front end and data getting deleted.
* **IsFull operation:** This operation is used to check if queue is full or not. As we are using array to implement queue we check whether the rear pointer is equal to MAXSIZE (size of Queue), if that is the case then user will get a message queue full. If queue is not full then user can insert new element.
* **IsEmpty operation:** This operation is used to check if queue is empty or not. This is done by checking whether the front pointer is less than MIN or greater than rear. If the queue is empty then user will get a queue empty message else user can delete the element from the queue.

When user performs these operations we can clearly see the insertion and deletion of elements into queue along with movement of FRONT and REAR pointer accordingly.

## 2.3.4Exit

This project includes an exit button that user can use to exit from the project once they are performing all the operation on both stack and queue.

# CHAPTER 3

**METHODOLOGY AND RESULTS**

**IMPLEMENTATION**

## 3.1 Description of Implementation Modules

In this project we have created structure of Stack and Queue using OpenGL functional API. We have taken the help of in-built functions present in the header file. To provide functionality to our project we have written sub functions. These functions provide us the efficient way to design the project. In this chapter we are describing the functionality of our project using these functions.

## 3.2 List of Implementation Functions

**void draw\_rectangle();** this function is used to create rectangle for Stack and Queue.

**void bg\_color();**

This function is used for background color of the project. **void displaying\_stack();**

This function is used to display structure of stack.

**void displaying\_simple\_queue();**

This function is used to display the structure of Queue. **void stack();**

This function is used to create Stack.

**void push();**

This function is used to push elements into Stack.

**void pop();**

This function is used to pop elements from Stack. **void simple\_queue();**

This function is used to create a Queue.

**void enqueuer();**

This function is used to insert element into Queue.

**void dequeue();**

This function is used to delete element from Queue.

**Void drawstring ();**

Draws the specified text string at the specified location.

**void mouse\_callback\_function();**

This function is used to set mouse to perform desired operation.

**Void Keyboard\_callback\_function();**

This function is used to set keys to perform desired operation.

**void Reshape\_callback\_function();**

This function is used to set the sizing of the window.

## 3.3 Description of the functions

**Main();**

The execution of the program starts from the main ().

**glutInit();**

Initializes GLUT, The argument from main are passed in and can be used by the application.

**glClear ();**

Clears buffers to preset values.

**glClearColor ();**

Specifies clear values for the color buffers. **glColor();**

Set the current color.**glLoadIdentity ();**

Replaces the current matrix with the identity matrix.

**glutInitDisplayMode();**

This function sets the display mode.

**glutInitWindowSize();**

This function Sets display-window width and height.

**glutCreateWindow();**

This function creates the display window. **glutDisplayFunc();**

Invokes a function to create a picture within the current display window.

**glutMainLoop();**

Causes the program to enter an event processing loop. **myinit();**

This function is defined to initialize the window parameters.

**glutReshapeFunc();**

Register callback handler for window re-size event. **glutKeyboardFunc();**

Register callback handler for special-key event. **glutMouseFunc();**

Register callback handler for mouse event.

**CHAPTER 4**

# TESTING

## 4.1 Testing

The primary purpose of testing is to detect software failures so that defects may be discovered and corrected. The scope of software testing often includes examination of codes as well as execution of that code in various environments.

**Table 4.1** Testing Table

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST CASE** | **RESULT EXPECTED** | **RESULT OBTAINED** | **REMARKS** |
| Stack of desired size is selected | Stack of given size appears | Stack of given size appears | Successful |
| Push operation | Elements inserted appears in the stack | Elements inserted by user appears in the stack | Successful |
| Pop operation | Elements deleted is removed from stack | Elements deleted is removed from stack | Successful |
| Isfull operation | Stack is full message appears | Stack full message appears | Successful |
| IsEmpty operation | Stack is empty message appears | Stack is empty message appears | Successful |
| Queue of desired size is selected | Queue of given size appears | Queue of given size appears | Successful |
| Enqueue operation | Elements inserted appears in the Queue | Elements inserted appears in the Queue | Successful |
| Dequeue operation | Elements deleted is removed from Queue | Elements deleted is removed from Queue | Successful |
| Overflow | Overflow message appears | Overflow message appears | Successful |
| operation |  |  |  |
| Underflow operation | Underflow message appears | Underflow message appears | Successful |

## 4.2 User Interface Testing

Interface testing is done to cover all the functionalities of the system and ensure that this project’s graphical user interface meets its specification. We test the interface to perform, such as the movements of the virus and antivirus and also swapping of displays.

## 4.3 System Testing

After testing the user interface, we can click into the system to test the program. We need to execute a number of times to make sure that the system does not have any bugs during execution. We check the compliance with the specified requirements.

## 4.4 Performance

The overall performance of the result is efficient and as expected. We need to test in different Windows OS and repeat test the user interface and system testing in these operating systems.

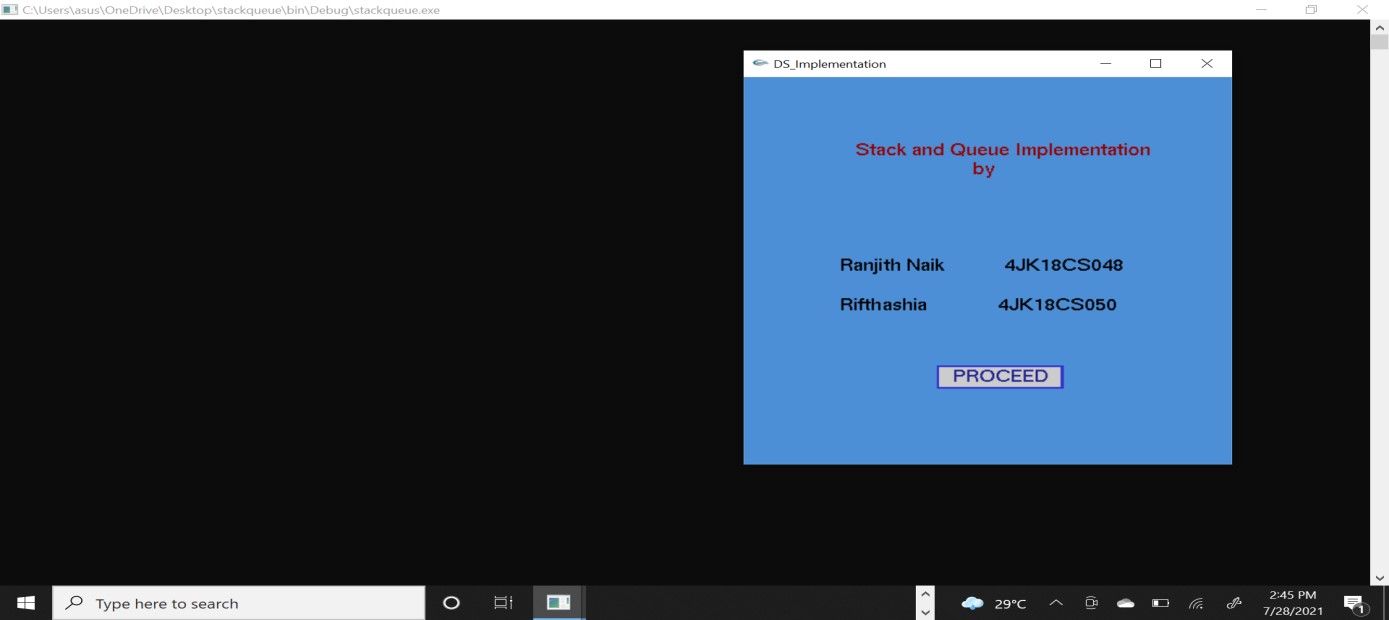
The simulation has no delays and the total simulation time approximates to 1 minute.

**CHAPTER 5**

# RESULTS

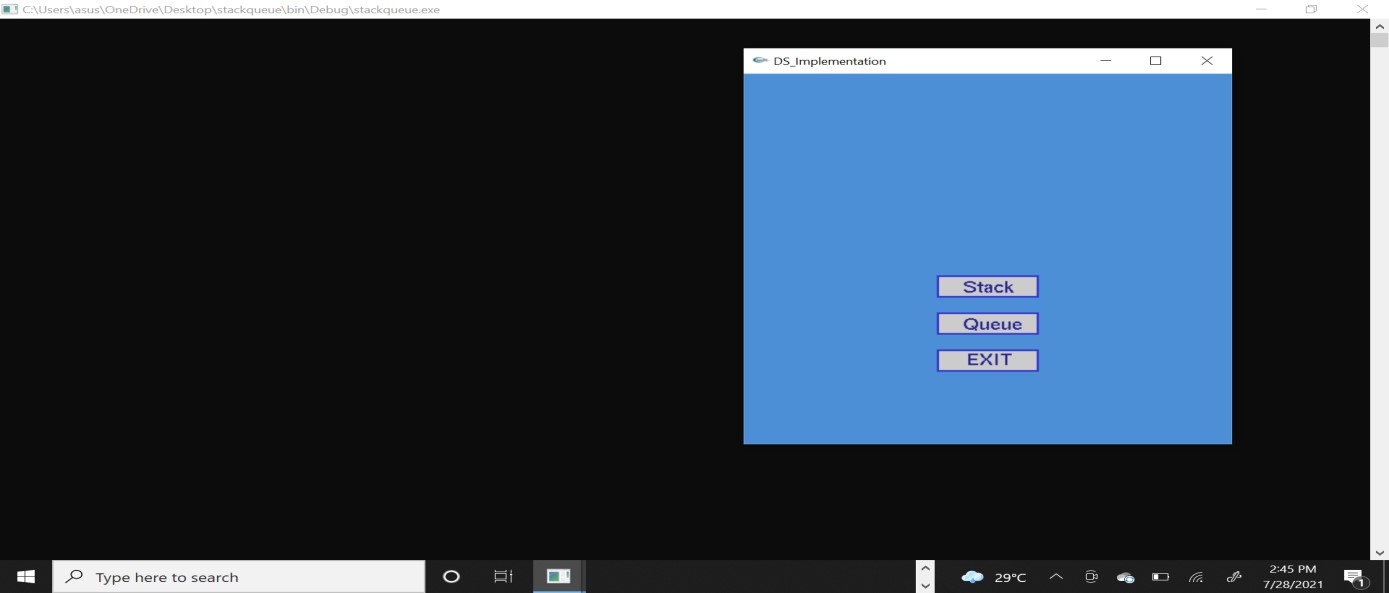
## 5.1 Screenshots of Results

The main window contains the Title of the project and the names and USN of the student who have created the project. It also contains a PROCEED button to go to the next window.



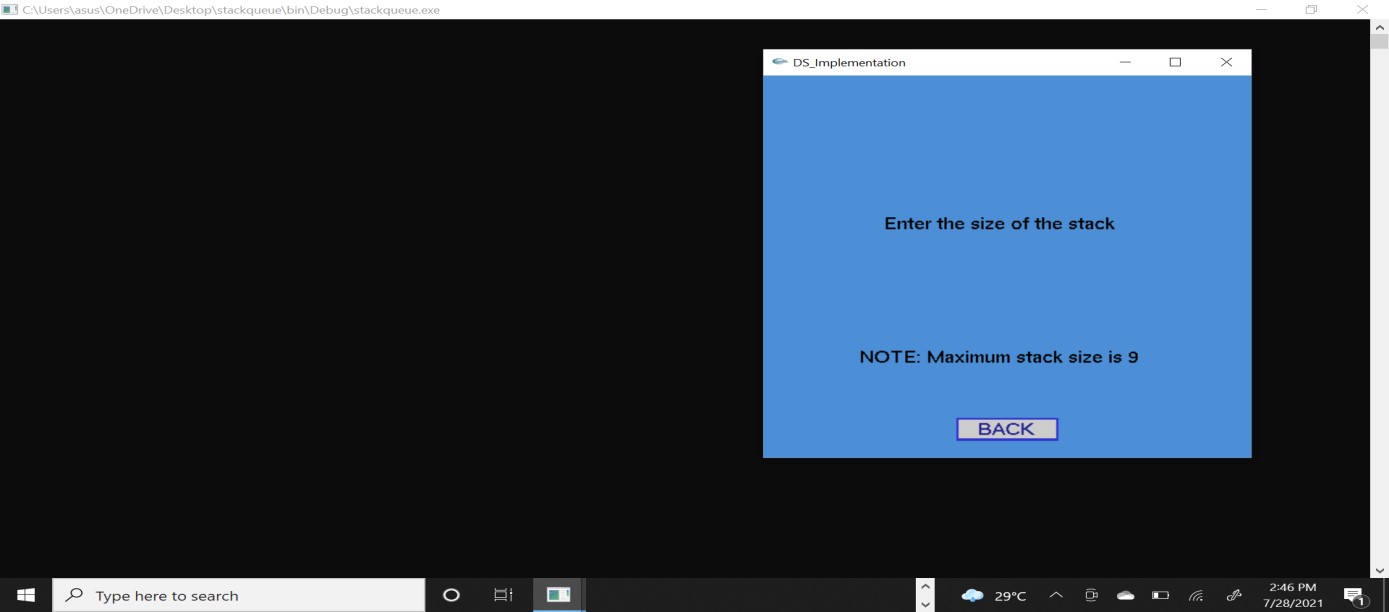
**Figure 5.1:** Main Window

This Window contains the options to be selected i.e Stack, Queue or Exit. User can make the choices.



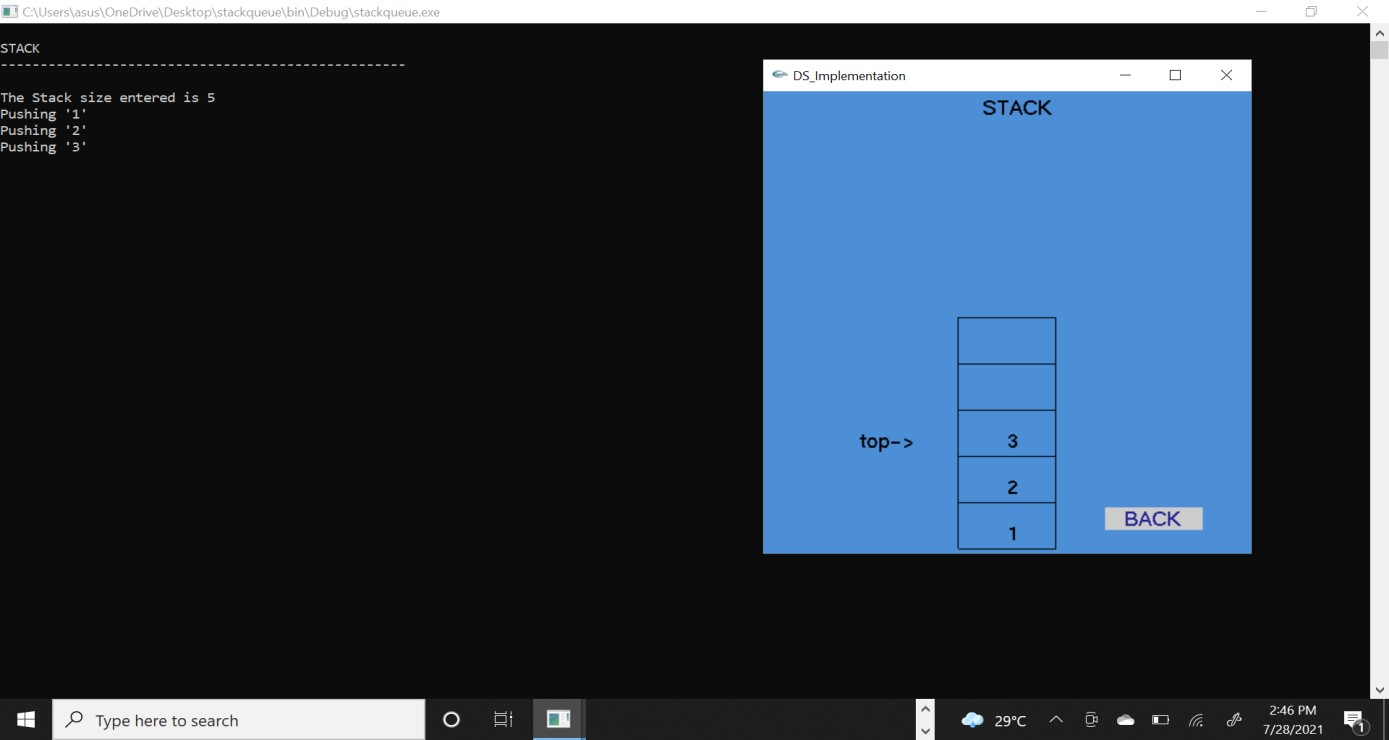
**Figure 5.2:** Selection Window

Figure shows the screenshot of the Stack size selecting window, user can select any desired size for the stack in this window, it is specified to the user that stack size can be between 0 and 9.



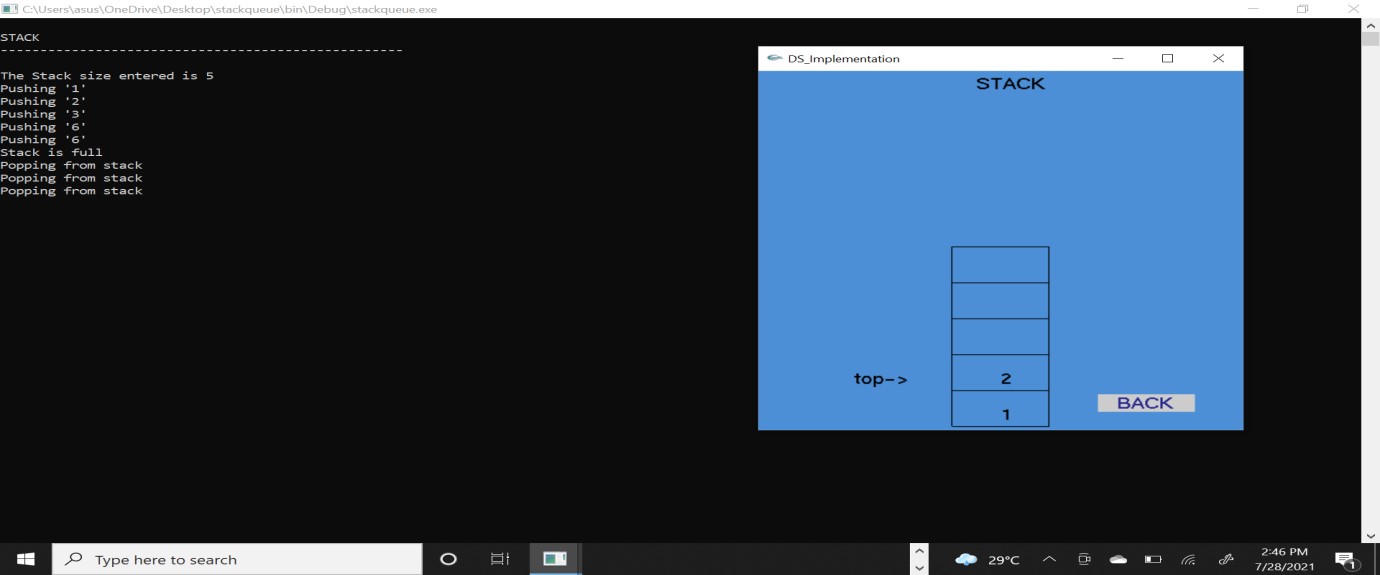
**Figure 5.3:** Stack Size Window

Once the stack of desired size appears user can start pushing any elements to the stack**.** There is also a back button so that user can go back and choose a different size if needed.



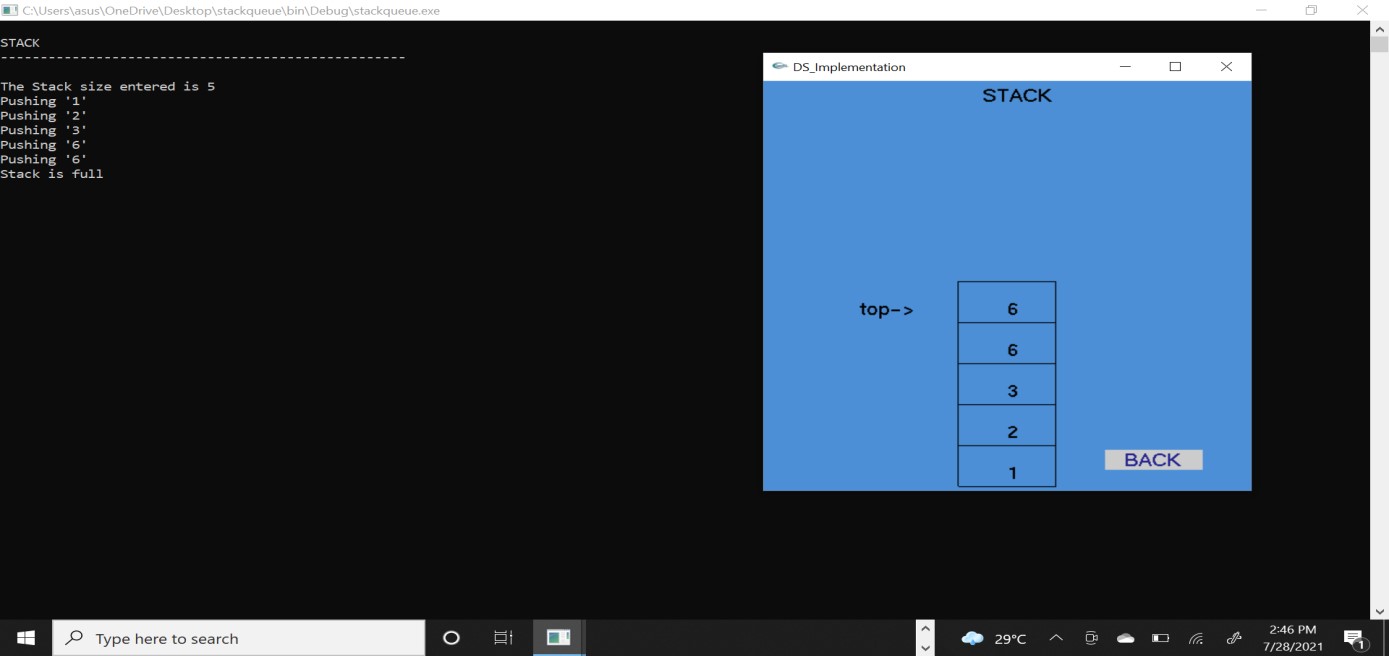
**Figure 5.4:** Push Operation

Once the stack of desired size appears user can start poping any elements from the stack**,** simultaneously top pointer is decremented. There is also a back button so that user can go back and choose a different size if needed.



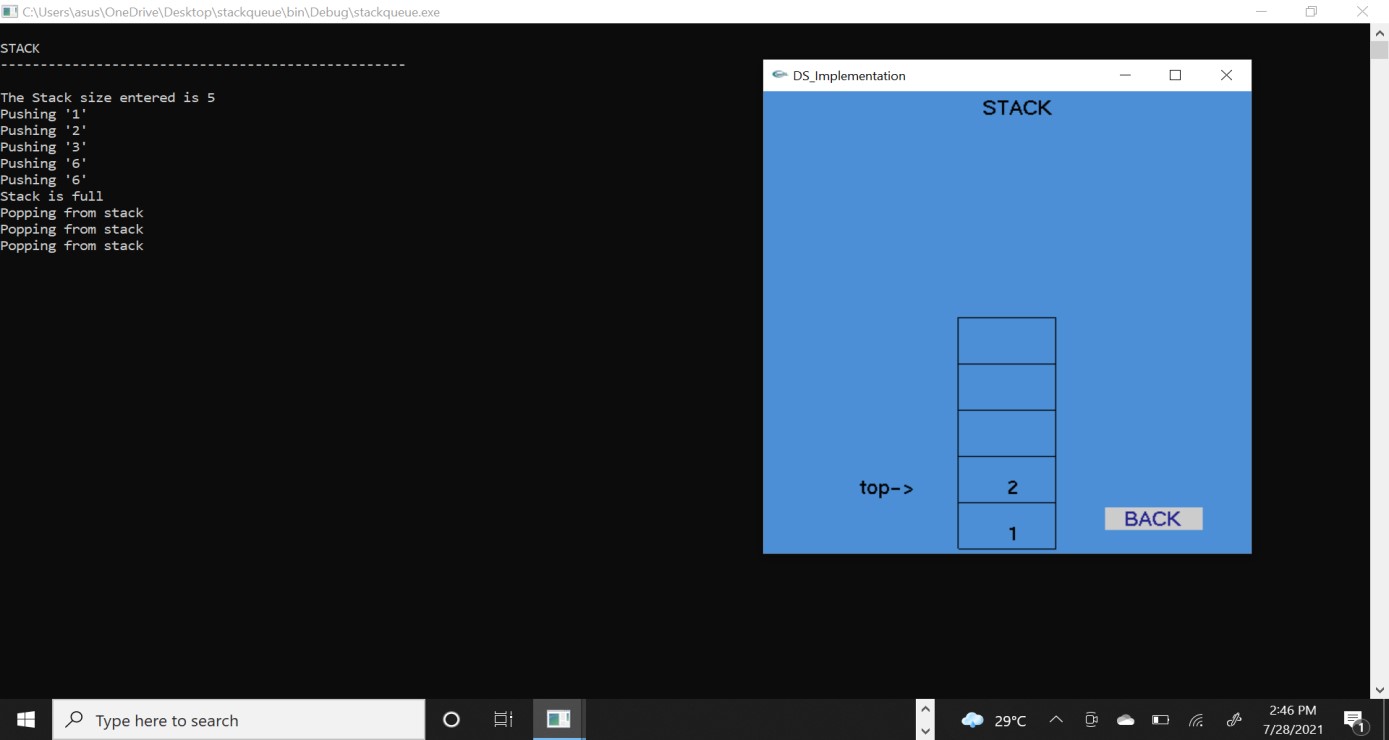
**Figure 5.5:** Pop Operation

As the user enters the element into the stack, the top pointer is incremented. Once the stack is full user gets a Stack is full message, user can no longer push elements into the stack.



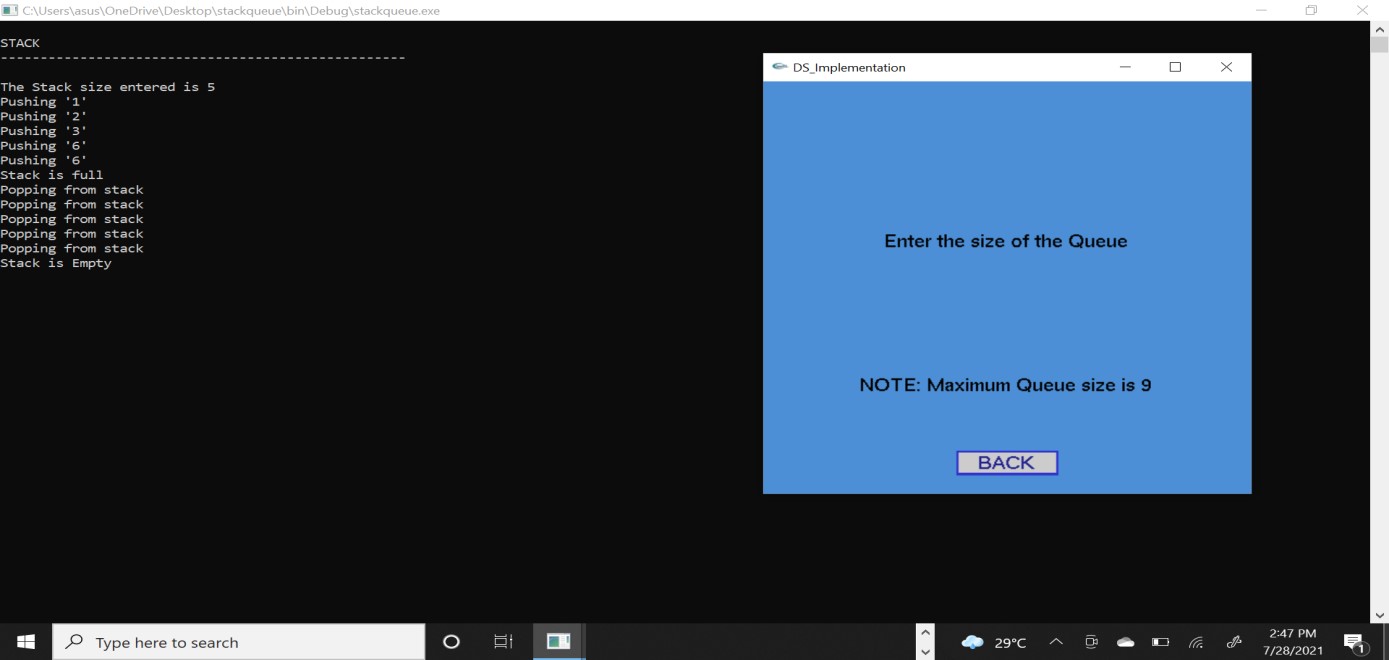
**Figure 5.6:** Stack is Full Condition

As the user deletes the element from the stack, the top pointer is decremented. Once the stack is empty user gets a Stack is empty message, user can no longer pop elements from the stack.



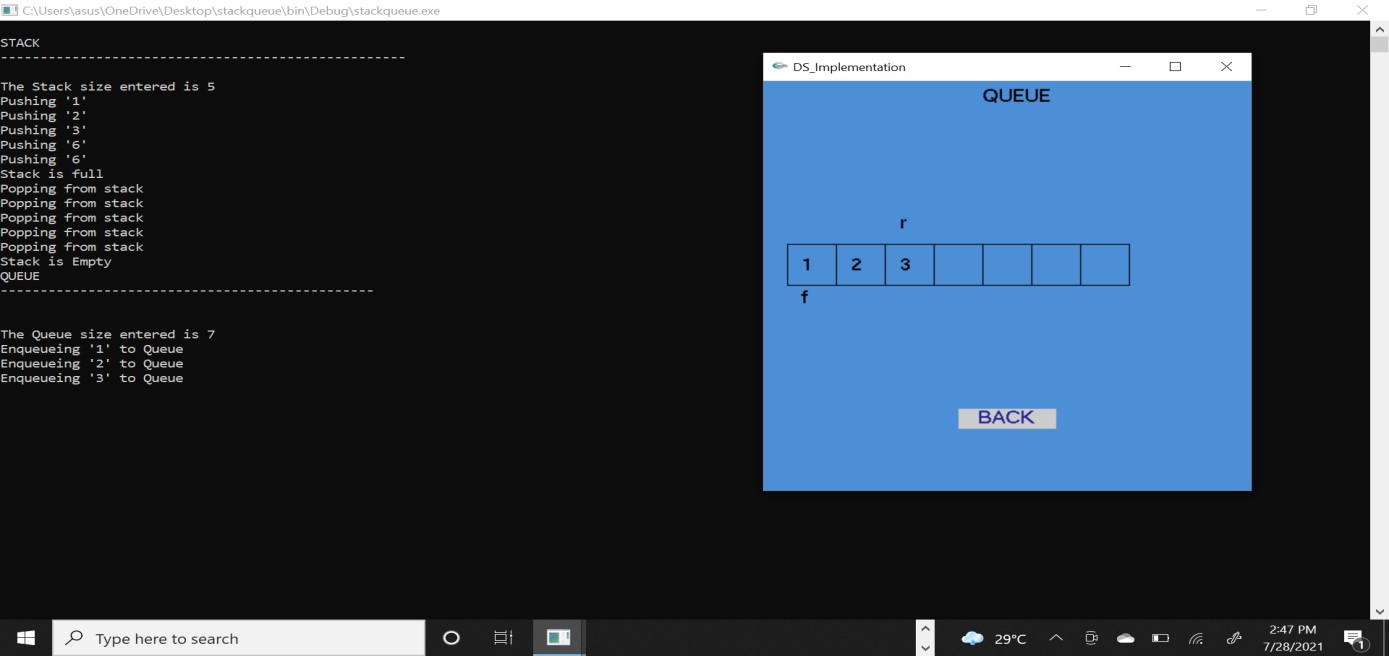
**Figure 5.7:** Stack is Empty Condition

User can select any desired size for the Queue in this window, it is specified to the user that stack size can be between 0 and 9.



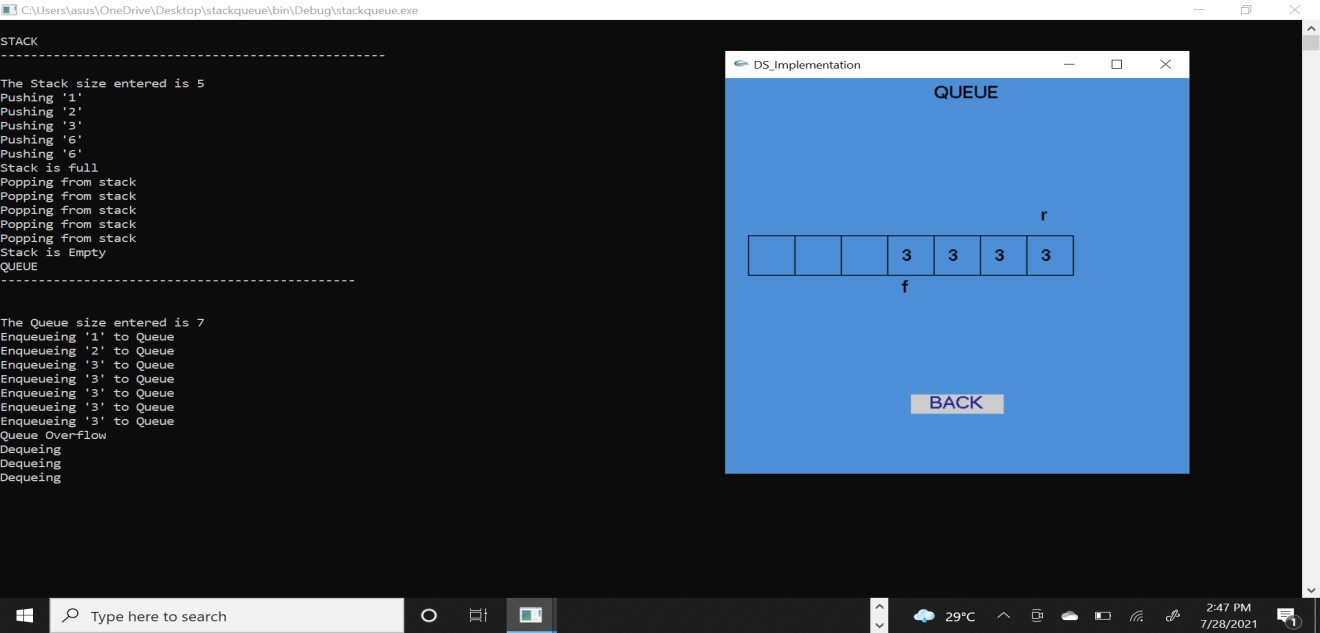
**Figure 5.8:** Queue Size Window

Once the Queue of desired size appears user can start inserting any elements into the Queue**.** There is also a back button in the bottom that user can go back and choose a different size if needed.



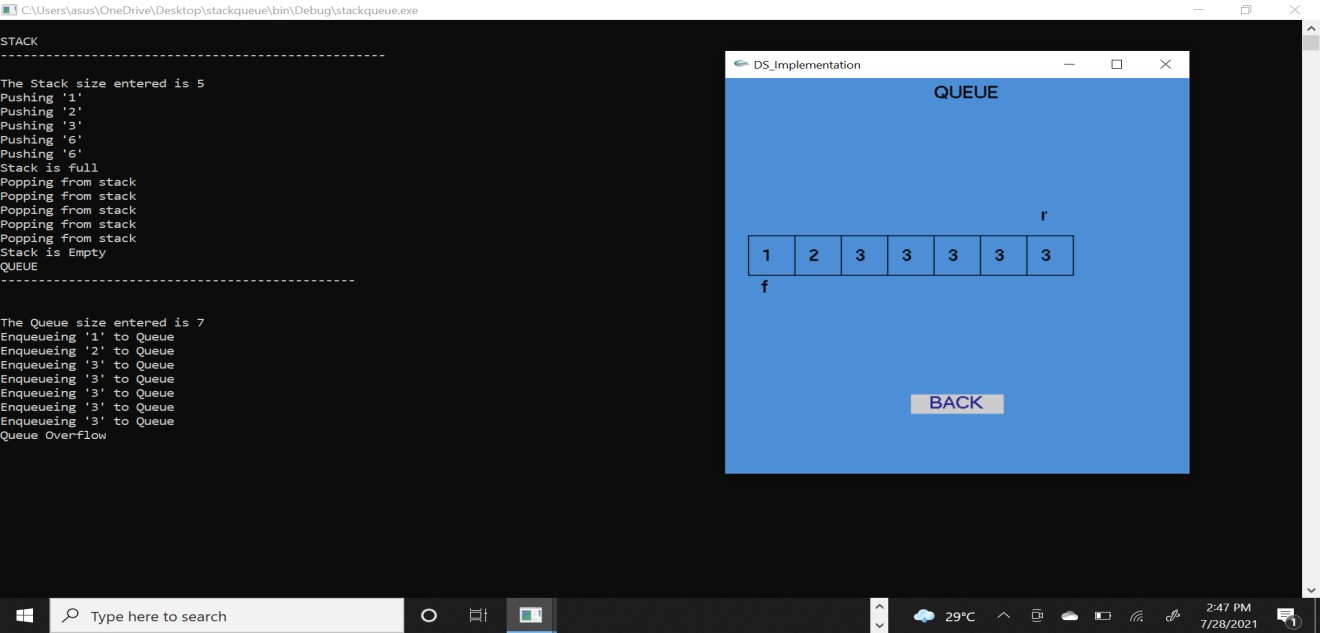
**Figure 5.9:** Enqueueing Operation in Queue

Once the Queue of desired size appears user can start deleting any elements from the Queue**.** As the elements are deleted the front pointer is incremented**.** There is also a back button so that user can go back and choose a different size if needed.



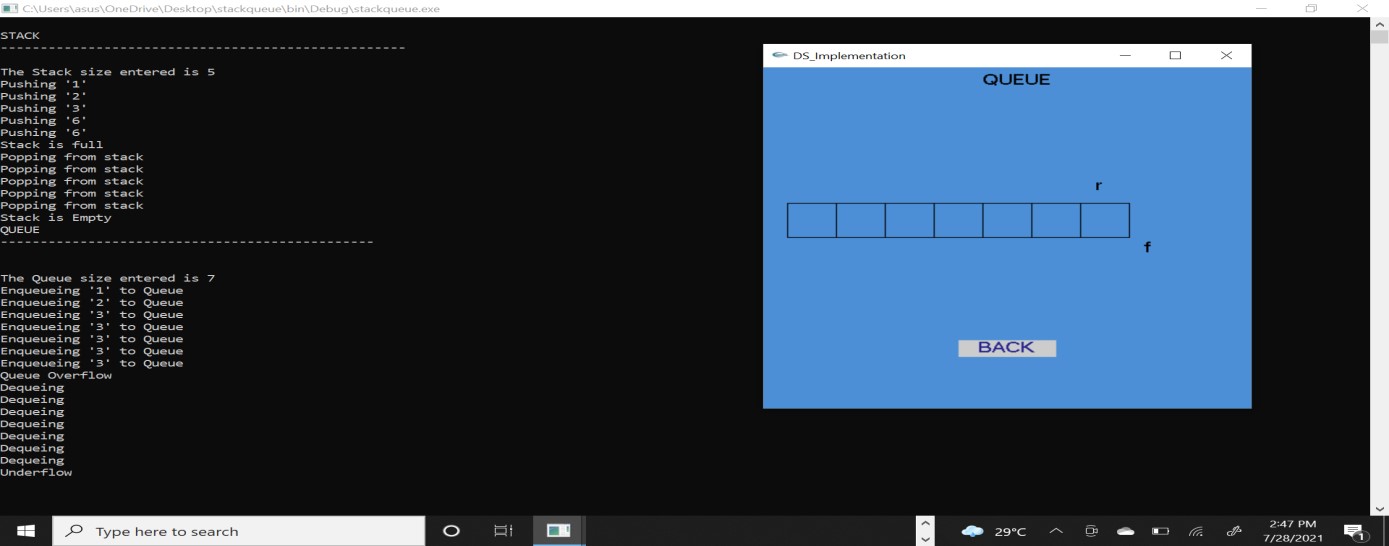
**Figure 5.10:** Dequeueing Operation in Queue

As the user enters the element into the Queue, the rear pointer is incremented. Once the Queue is full user gets a Queue Overflow message, user can no longer insert elements into the Queue.



**Figure 5.11:** Overflow Condition

As the user deletes the element from the Queue, the front pointer is decremented. Once the Queue is empty user gets a Queue Underflow message, user can no longer delete elements from the Queue.



**Figure 5.12:** Underflow Condition

**CHAPTER 6**

# CONCLUSIONS AND FUTURE ENHANCEMENTS

## 6.1 Conclusion

The project Stack and Queue Implementation has been successfully implemented using OpenGL and C++. The illustration of graphical principles and OpenGL features are included and application program is efficiently developed. This project is an effort in the development of a Graphical Software package which is the building block of graphical application. And it also helped us to understand various inbuilt functions of OpenGL.

The main aim of developing this project was to design a simple program using OpenGL application by applying the skills that we learnt in earlier classes and implement it, which helped improve our knowledge on both Computer graphics and algorithms. This project shows a visual representation of the basic operations that can be performed in Stack and Queue data structure.

So that students can have a clear picture regarding the working of these data structure.

## 6.2 Future Enhancement

* We can add a trigger warning for the user as the Stack or Queue is about to be full or empty.
* We could also add functions of Double ended Queue, Circular Queue etc.

## REFERENCES

1. OpenGL functions that we have used in the Project: G. Reina, T. Muller and T. Ertl,

“Incorporating Modern OpenGL into Computer Graphics Education,” in IEEE Computer Graphics and Applications, vol. 34, no. 4, pp.16-21, july-aug. 2014

1. OpenGL along with Computer Graphics: Interactive Computer Graphics a Top-Down Approach with OpenGL-Edward Angel, 6 Edition, Addison-wesley, 2008.
2. Computer Graphics reference: Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd / 4th Edition, Pearson Education, 2011.
3. Stack functions that we have referred for the project: https://www.geeksforgeeks.org/stackdata-structure/
4. Queue functions that we have referred for the project: https://www.geeksforgeeks.org/queue- data-structure/