**Chapter 1**

**Preamble**

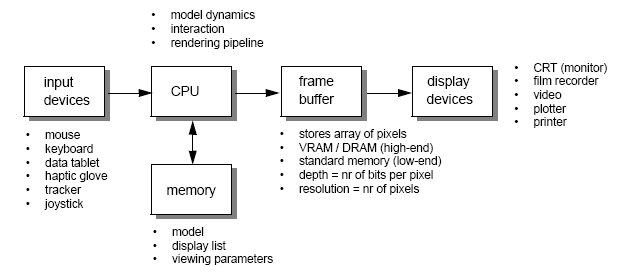
* 1. **Introduction to Computer Graphics**

Computer graphics are pictures and films created using computers. Usually, the term refers to computer-generated image data created with help from specialized graphical hardware and software. It is a vast and recent area in computer science. The phrase was coined in 1960, by computer graphics researchers Verne Hudson and William Fetter of Boeing. It is often abbreviated as CG, though sometimes erroneously referred to as computer-generated imagery (CGI).Some topics in computer graphics include [user interface design](https://en.wikipedia.org/wiki/User_interface_design), [sprite graphics](https://en.wikipedia.org/wiki/Sprite_(graphics)), [vector graphics](https://en.wikipedia.org/wiki/Vector_graphics), [3D modelling](https://en.wikipedia.org/wiki/3D_modeling), [shaders](https://en.wikipedia.org/wiki/Shader), [GPU](https://en.wikipedia.org/wiki/GPU) design, [implicit surface](https://en.wikipedia.org/wiki/Implicit_surface) visualization with [ray tracing](https://en.wikipedia.org/wiki/Ray_tracing_(graphics)), and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), among others. The overall methodology depends heavily on the underlying sciences of [geometry](https://en.wikipedia.org/wiki/Geometry), [optics](https://en.wikipedia.org/wiki/Optics), and physics. Computer graphics is responsible for displaying art and image data effectively and meaningfully to the user. It is also used for processing image data received from the physical world. Computer graphic development has had a significant impact on many types of media and has revolutionized [animation](https://en.wikipedia.org/wiki/Animation), [movies](https://en.wikipedia.org/wiki/Movies), [advertising](https://en.wikipedia.org/wiki/Advertising), [video games](https://en.wikipedia.org/wiki/Video_game), and [graphic design](https://en.wikipedia.org/wiki/Graphic_design) generally [1].

The term computer graphics has been used in a broad sense to describe "almost everything on computers that is not text or sound". Typically, the term computer graphics refers to several different things: [1]

* the representation and manipulation of image data by a computer
* the various [technologies](https://en.wikipedia.org/wiki/Technologies) used to create and manipulate images
* the sub-field of [computer science](https://en.wikipedia.org/wiki/Computer_science) which studies methods for digitally synthesizing and manipulating visual content, see [study of computer graphics](https://en.wikipedia.org/wiki/Computer_graphics_(computer_science))

Today, computer graphics is widespread. Such imagery is found in and on television, newspapers, weather reports, and in a variety of medical investigations and surgical procedures. A well-constructed [graph](https://en.wikipedia.org/wiki/Chart) can present complex statistics in a form that is easier to understand and interpret. In the media "such graphs are used to illustrate papers, reports, theses", and other presentation material.

Many tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. As technology has improved, [3D computer graphics](https://en.wikipedia.org/wiki/3D_computer_graphics) have become more common, but [2D computer graphics](https://en.wikipedia.org/wiki/2D_computer_graphics) are still widely used. Computer graphics has emerged as a sub-field of [computer science](https://en.wikipedia.org/wiki/Computer_science) which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like [information visualization](https://en.wikipedia.org/wiki/Information_visualization), and [scientific visualization](https://en.wikipedia.org/wiki/Scientific_visualization) more concerned with "the visualization of [three dimensional](https://en.wikipedia.org/wiki/Three-dimensional_space) phenomena (architectural, meteorological, medical, [biological](https://en.wikipedia.org/wiki/Biological_Data_Visualization), etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component". [5]

**Figure 1.1**: Graphics Pipeline

* 1. **History of Computer Graphics**

The phrase Computer Graphics was coined in 1960 by William Fetter, a graphic designer for Boeing. The field of Computer Graphics developed with the emergence of computer graphics hardware. Early projects like the Whirlwind and SAGE projects introduced the CRT as a viable display and interaction interface and introduced the light pen as an input device.[1]

Further advances in computing led to greater advancements in interactive computer graphics. In 1959, the TX-2 computer was developed at MIT's Lincoln Laboratory. A light pen could be used to draw sketches on the computer using Ivan Sutherland’s revolutionary Sketchpad software.[1]

Also, in 1961 another student at MIT, Steve Russell, created the first video game, Spacewar!E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-gyro gravity attitude control system" in 1963. In this computer-generated film, Zajac showed how the attitude of a satellite could be altered as it orbits the Earth. Many of the most important early breakthroughs in computer graphics research occurred at the University of Utah in the 1970s.

The first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image.

Graphics and application processing were increasingly migrated to the intelligence in the workstation, rather than continuing to rely on central mainframe and mini-computers. 3D graphics became more popular in the 1990s in gaming, multimedia and animation. Computer graphics used in films and video games gradually began to be realistic to the point of entering the uncanny valley. Examples include the later Final Fantasy games and animated films like The Polar Express.[1]

* 1. **Applications of Computer graphics**

The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. The applications of computer graphics are many and varied. We can however divide them into four major areas [6]

* Display of information: More than 4000 years ago, the Babylonians developed floor plans of buildings on stones. Today, the same type of information is generated by architects using computers. Over the past 150 years, workers in the field of statistics have explored techniques for generating plots. Now, we have computer plotting packages. Supercomputers now allow researchers in many areas to solve previously intractable problems. Thus, Computer Graphics has innumerable applications.
* Design: Professions such as engineering and architecture are concerned with design. Today, the use of interactive graphical tools in CAD, in VLSI circuits, characters for animation have developed in a great way.
* Simulation and animation: One of the most important uses has been in pilots’ training. Graphical flight simulators have proved to increase safety and reduce expenses. Simulators can be used for designing robots, plan it’s path, etc. Video games and animated movies can now be made with low expenses.
* User interfaces: Our interaction with computers has become dominated by a visual paradigm. The users’ access to internet is through graphical network browsers. Thus Computer Graphics plays a major role in all fields.

* 1. **Open Graphics Library (OpenGL)**

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that are used to specify the objects and operations needed to produce interactive three-dimensional applications. OpenGL is designed as a streamlined hardware-independent interface to be implemented on many different hardware platforms. [2]

These are certain characteristics of OpenGL:

* OpenGL is a better documented API.
* OpenGL is much easier to learn and program.
* OpenGL has the best demonstrated 3D performance for any API.

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it's possible for the API to be implemented entirely in software, it's designed to be implemented mostly or entirely in hardware.

In addition to being language-independent, OpenGL is also platform-independent. The specification says nothing on the subject of obtaining, and managing, an OpenGL context, leaving this as a detail of the underlying windowing system. For the same reason, OpenGL is purely concerned with rendering, providing no APIs related to input, audio, or windowing.

OpenGL is an evolving API. New versions of the OpenGL specification are regularly released by the Khronos Group, each of which extends the API to support various new features. In addition to the features required by the core API, GPU vendors may provide additional functionality in the form of extensions. Extensions may introduce new functions and new constants and may relax or remove restrictions on existing OpenGL functions. Vendors can use extensions to expose custom APIs without needing support from other vendors or the Khronos Group as a whole, which greatly increases the flexibility of OpenGL. All extensions are collected in, and defined by, the OpenGL Registry. [2]

* 1. **OpenGL Utility Library (GLU)**

The OpenGL Utility Library (GLU) was a computer graphics library for OpenGL.It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides. It is usually distributed with the base OpenGL package. GLU is not implemented in the embedded version of the OpenGL package, OpenGL ES. [5]

Among these features are mapping between screen- and world-coordinates, generation of texture  mipmaps , drawing of quadric surfaces, NURBS, tessellation of polygonal primitives, interpretation of OpenGL error codes, an extended range of transformation routines for setting up viewing volumes and simple positioning of the camera, generally in more human-friendly terms than the routines presented by OpenGL. It also provides additional primitives for use in OpenGL applications, including spheres, cylinders and disks.

All GLU functions start with the glu prefix. An example function is gluOrtho2D which defines a two-dimensional orthographic projection matrix.

* 1. **OpenGL Utility Toolkit (GLUT)**

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. GLUT also has some limited support for creating pop-up menus. [2]

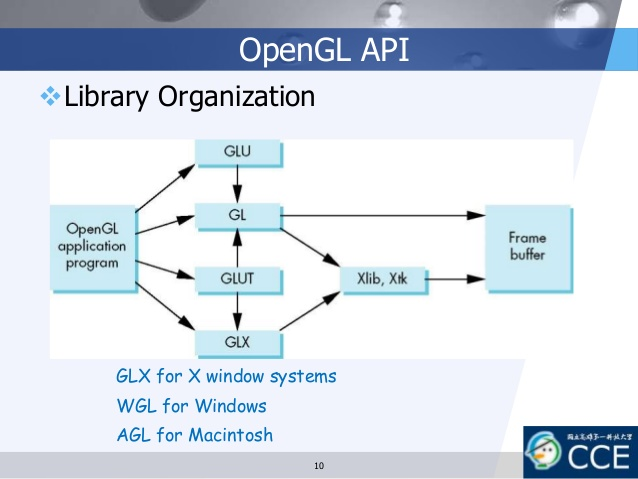
GLUT was written by Mark J. Kilgard, author of OpenGL Programming for the X Window System and The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics, while he was working for Silicon Graphics Inc. [2]

The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross-platform) and to make learning OpenGL easier. Getting started with OpenGL programming while using GLUT often takes only a few lines of code and does not require knowledge of operating system–specific windowing APIs. All GLUT functions start with the glut prefix , for example, glutPostRedisplay marks the current window as needing to be redrawn. [5]

The toolkit supports:

* Multiple windows for OpenGL rendering and callback driven event processing
* Sophisticated input devices
* An 'idle' routine and timers
* A simple, cascading pop-up menu facility
* Utility routines to generate various solid and wire frame objects
* Support for bitmap and stroke fonts

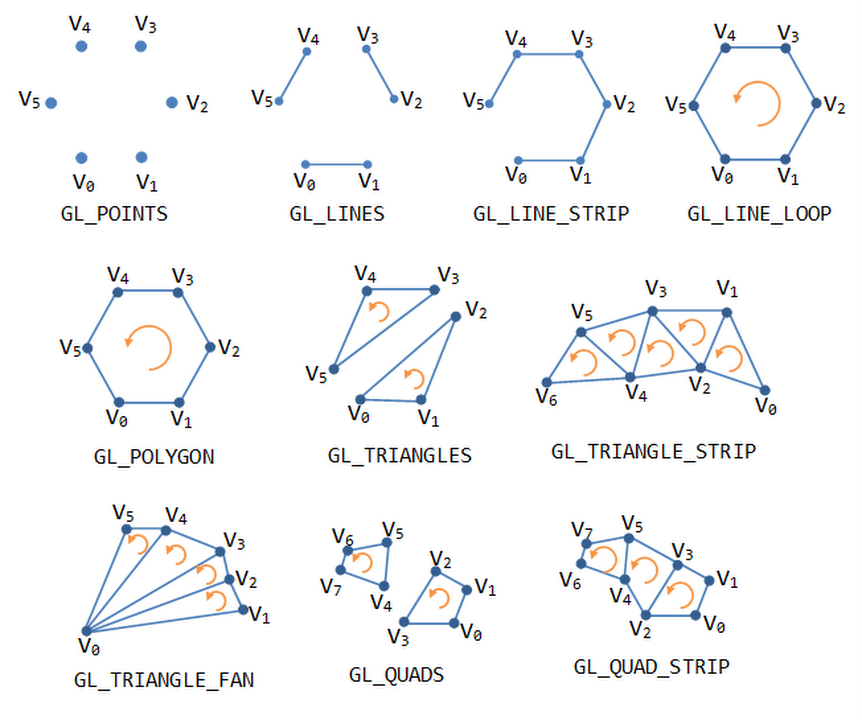
Some of the more notable limitations of the original GLUT library by Mark Kilgard include:

* The library requires programmers to call glutMainLoop(), a function which never returns. This makes it hard for programmers to integrate GLUT into a program or library which wishes to have control of its own event loop. A common patch to fix this is to introduce a new function, called glutCheckLoop() (macOS) or glutMainLoopEvent() (FreeGLUT/OpenGLUT), which runs only a single iteration of the GLUT event loop. Another common workaround is to run GLUT's event loop in a separate thread, although this may vary by operating system, and also may introduce synchronization issues or other problems: for example, the macOS GLUT implementation requires that glutMainLoop() be run in the main thread. [2]
* The fact that glutMainLoop() never returns also means that a GLUT program cannot exit the event loop. FreeGLUT fixes this by introducing a new function, glutLeaveMainLoop().
* The library terminates the process when the window is closed; for some applications this may not be desired. Thus, many implementations include an extra callback, such as glutWMCloseFunc().

**Figure 1.2**: Library Organization of OpenGL APIs

* 1. **Applications of OpenGL**
* OpenGL (Open Graphics Library) is a cross-language, multi-platform API for rendering 2D and 3D computer graphics.
* The API is typically used to interact with a GPU, to achieve hardware-accelerated rendering.
* It is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation.
* It is also widely used in the development of video games for different platforms such as PC, consoles or smart phones.
  1. **OpenGL Primitives**

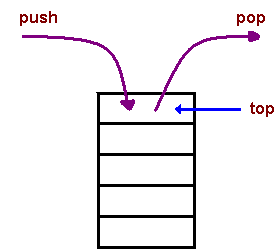
OpenGL supports two classes of primitives:

* Geometric Primitives: Geometric primitives are specified in the problem domain and include points, line segments, polygons, curves and surfaces.
* Image (Raster) Primitives: Raster primitives, such as arrays of pixels pass through a separate parallel pipeline on their way to the frame buffer.

**Figure 1.3**: OpenGL Geometric Primitives

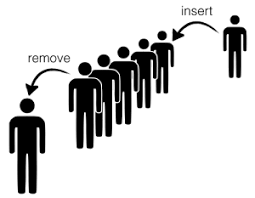
* 1. **Introduction to Stack and Queue**

A data structure is a method of organizing information. These structures include files, lists, arrays, trees, records and tables. Stack is a special type of data structure (linear data structure) where elements are inserted from one end and elements are deleted from the same end. Using this approach, the **L**ast element **I**nserted is the **F**irst element to be deleted **O**ut and hence, stack is also called **L**ast **I**n **F**irst **O**ut (**LIFO**) data structure.



**Figure 1.4**: Stack

Queue is a special type of data structure (an ordered collection of items) where elements are inserted from one end and elements are deleted from the other end. The end at which new elements are added is called the rear and the end from which elements are deleted is called the front. Using this approach, the **F**irst element **I**nserted is the **F**irst element to be deleted **O**ut, and hence, queue is also called **F**irst **I**n **F**irst **O**ut(**FIFO**) data structure.



**Figure 1.5**: Queue

Stack is an important data structure which has got a wide range of applications in its field, now it's importance lies on the fact that it follows something called as LIFO/last in first out. Now importance of an entity comes to us only when we start using it either directly or indirectly. One such indirect use of stack in our daily life is in function call within a function. It's an interesting fact to notice that when we call a function within a function repeatedly the last function called get executed first.

Queues are related to ordered lists. With the queue, the new pieces of data are placed at the rear of the data structure, and the deletions are placed at the front. The first piece of data entered into the data structure is the first piece removed from the structure. With queues, data does not remain in the data structure for as long as with stacks. Queues can be compared to lines at the store, where the first person in line is the first person to receive a service.

This is a live project which is developed for implementing Stack and queue. A stack is a container of objects that are inserted and removed according to last-in-first-out (LIFO) principle. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. Push adds an item to the top of the stack, pop removes the item from the top. Queue Structure is the crucial element of a queuing system, as it shows the queue discipline, which means the order in which the customers are picked from the queue for the service. Simply, the way the customer is selected from the waiting line for service is shown by the queue structure. The platform used to develop this project is visual studio 2017 which is linked to OpenGL . The operating system used is windows 10 pro, 64 bits. The language in which the source code is written to develop this project is “C++” language.

Basic operations that can be performed on stack and queue are:

* Creation
* Insertion (Queue)
* Deletion (Queue)
* Push (Stack)
* Pop (Stack)
* Display

Enqueue and Dequeue tend to be operations on a queue, a data structure that does exactly what it sounds like it does. You enqueue items at one end and dequeue at the other, There are variations of queues such as double-ended ones where you can enqueue and dequeue at either end but the vast majority would be the simpler form. Enqueue is used to add an object to the end of the Queue The value can null and if the Count is less than the capacity of the internal array, this method is an O(1) operation. If the internal array needs to be reallocated to accommodate the new element, this method becomes an O(n) operation, where n is Count.

[Deque or Double Ended Queue](http://en.wikipedia.org/wiki/Double-ended_queue) is a generalized version of [Queue data structure](http://quiz.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/)that allows insert and delete at both ends.

Mainly the following four basic operations are performed on queue:

* insertFront(): Adds an item at the front of Deque.
* insertLast(): Adds an item at the rear of Deque.
* deleteFront(): Deletes an item from front of Deque.
* deleteLast(): Deletes an item from rear of Deque.
* getFront(): Gets the front item from queue.
* getRear(): Gets the last item from queue.
* isEmpty(): Checks whether Deque is empty or not.
* isFull(): Checks whether Deque is full or not.
  1. **Objectives**

The basic objective behind developing this project “STACK AND QUEUE” is to properly understand how and where the operations of stack and queue are used. Such as:-

* Arithmetic expression evaluation
* Evaluating a postfix expression
* Converting from infix to postfix.
* Serving requests of a single shared resource (printer, disk, CPU), transferring data asynchronously (data not necessarily received at same rate as sent) between two processes (IO buffers), e.g., pipes, file IO, sockets.
* Buffers on MP3 players and portable CD players, iPod playlist. Playlist for jukebox - add songs to the end, play from the front of the list.
* Interrupt handling: When programming a real-time system that can be interrupted (e.g., by a mouse click or wireless connection), it is necessary to attend to the interrupts immediately, before proceeding with the current activity.
* If the interrupts should be handles in the same order they arrive, then a FIFO queue is the appropriate data structure.
  1. **Organization of the Report**

Chapter 1 provides the information about the basics of computer graphics, the history of OpenGL, the major OpenGL libraries and about the basics of Stack and Queue. Chapter 2 provides the hardware and software specifications required by a system to run this project. Chapter 3 gives the idea of the project. Chapter 4 discusses about the algorithm used to develop the program and its actual implementation. Chapter 5 provides various screenshots of the program in run time. Chapter 6 concludes by giving directions for future enhancements.

* 1. **Summary**

The chapter discussed before is an overview about Computer graphics, its history, the various utility tools to develop this project and also a brief introduction on Stack and Queue. It even includes the Graphics pipeline diagram, the timing and the sender’s and receiver’s window diagrams. The scope of study and objectives of the project are mentioned clearly. The organization of the report has been pictured to increase the readability. Further, coming up chapters discuss about the overall working of the project.

**Chapter 2**

**System Specifications**

**2.1 Software Requirements**

Operating system : Windows 10

Compiler used : GCC

Programming language : C++ language

Editor : Visual Studio

Graphics library : GL and GLU / GLUT

**2.2 Hardware Requirements**

Processor : Intel I3/I5/I7

Processor speed : 1GHz or more

Hard disk : 40 GB or more

RAM size : 1 GB or more

Display : 800x600 or higher resolution display with 256 colours

Mouse : Standard serial mouse

Keyboard : Standard QWERTY keyboard

GPU : Intel HD Graphics 5000 or better

**Chapter 3**

**Stack and Queue**

There are many real-life examples of a stack. Consider the simple example of plates stacked over one another in a canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO/FILO order.



**Figure 3.1**: Basic idea of stack

 A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between [stacks](https://www.geeksforgeeks.org/stack-data-structure/)and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.



**Figure 3.2**: Basic idea of queue

The program is interactive, uses keyboard functions to select the options. It is also possible to select push, pop, insertion and deletion with the press of a key. It is also possible to switch between operations using specific keys.

The program showcases how the elements are pushed, poped, inserted, deleted, inserted from both the end of the queue and deleted from both the end of the queue.

**Chapter 4**

**System Design and Implementation**

**4.1 Introduction**

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

* 1. **Initialization**

Initialize the interaction with the windows. Initialize the display mode- double buffer and depth buffer. Initialize the various callback functions for drawing and redrawing, for mouse and keyboard interfaces. Initialize the window position and size and create the window to display the output.

* 1. **Flow of control**

The flow of control in the below flow chart is with respect to the Stack and Queue. For any of the program flow chart is compulsory to understand the program. We consider the flow chart for the texture project in which the flow starts from start and proceeds to the main function after which it comes to the initialization of call back functions and further it proceeds to mouse and keyboard functions, input and Finally, it comes to quit, the end of flow chart.

MAIN

INITIALIZE CALLBACK FUNCTIONS

MAIN SCREEN DISPLAYED

END OF INPUT

MOUSE INTERUPT

READING INPUT

KEYBOARD

DISPLAY OUTPUT

**Figure 4.1**: Project Design

* 1. **OpenGL APIs used/Built-in functions**
* **glRasterPos3f()**
  + - 1. Specifies the raster position for pixel operations.
* **glutBitmapCharacter()**
  + - 1. Renders a bitmap charater using OpenGL from the specified array of characters, and in the specified font style.
* **glutPostRedisplay()** 
  + - 1. Marks the current window as needing to be redisplayed.
* **glutTimerFunc( )**
  + - 1. Registers a timer callback to be triggered in a specified number of milliseconds.
* **glClearColor ()**
  + - 1. Specifies clear values for the color buffers.
* **glShadeModel ()**
  + - 1. Select flat or smooth shading. Specifies a symbolic value representing a shading technique. Accepted values are GL\_FLAT and GL\_SMOOTH.
* **glEnable()**
  + - 1. Enables the OpenGL capabilities, Specifies the conditions under which the pixels will be drawn.
* **glLightfv()**
  + - 1. Creates a new light source with specified parameter values.
* **gluQuadricDrawStyle()**
  + - 1. Specifies the draw style required for quadrics.
* **glPushMatrix()** and**glPopMatrix()**
  + - 1. Push and pop the current matrix stack.
* **glTranslatef() and glRotatef()**
  + - 1. Multiplies current matrix by Translation and Rotation matrix respectively.
* **glMatrixMode ()**
  + - 1. Specifies which matrix is the current matrix.
* **glLoadIdentity()**
  + - 1. Pushes the identity matrix to the top of the matrix stack.
* **gluLookAt()**
  + - 1. Defines a viewing transformation.
* **glutSwapBuffers()**
  + - 1. Swaps the buffers of the current window if double buffered.
* **glViewport()**
  + - 1. Sets the viewport.
* **glutInitDisplayMode ()**
  + - 1. Sets the initial display mode.
* **glutInitWindowSize ()** and **glutInitWindowPosition ()**
  + - 1. Set the initial window size and position respectively.
* **glutCreateWindow()**
  + - 1. Creates a top level window with the window name as specified.
* **glutAddMenuEntry()**
  + - 1. Adds a menu entry to the bottom of the current menu.
* **glutAttachMenu()**
  + - 1. Attaches a mouse button for the current window to the identifier of the current menu**.**
* **glutDisplayFunc()**
  + - 1. Sets the display callback for the current window.
* **glutReshapeFunc()**
  + - 1. Sets the reshape callback for the current window.
* **glutMainLoop()**
  + - 1. Enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

**4.5 Pseudo Codes/Algorithms**

* Push operation for Stack [3]

procedure push(stk : stack, x : item):

if stk.top = stk.maxsize:

report overflow error

else:

stk.items[stk.top] ← x

stk.top ← stk.top +

* Pop operation for Stack [3]

procedure pop(stk : stack):

if stk.top = 0:

report underflow error

else:

stk.top ← stk.top − 1

r ← stk.items[stk.top]

return r

* Insert operation for Queue [4]

procedure insert(data)

if queue is full

return overflow

endif

rear ← rear + 1

queue[rear] ← data

return true

end procedure

* Delete Operation for Queue [4]

procedure delete

if queue is empty

return underflow

end if

data = queue[front]

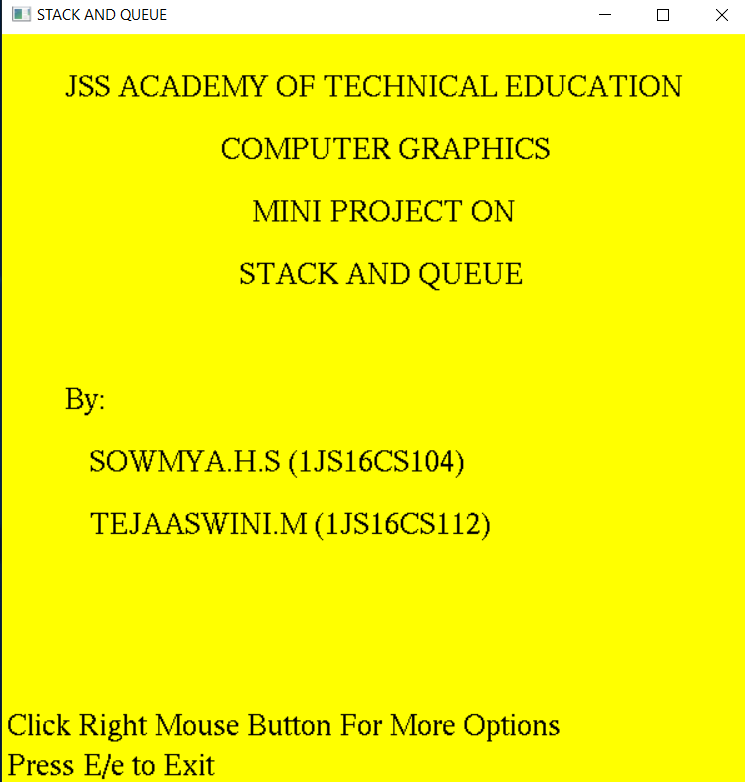
front ← front + 1

return true

end procedure

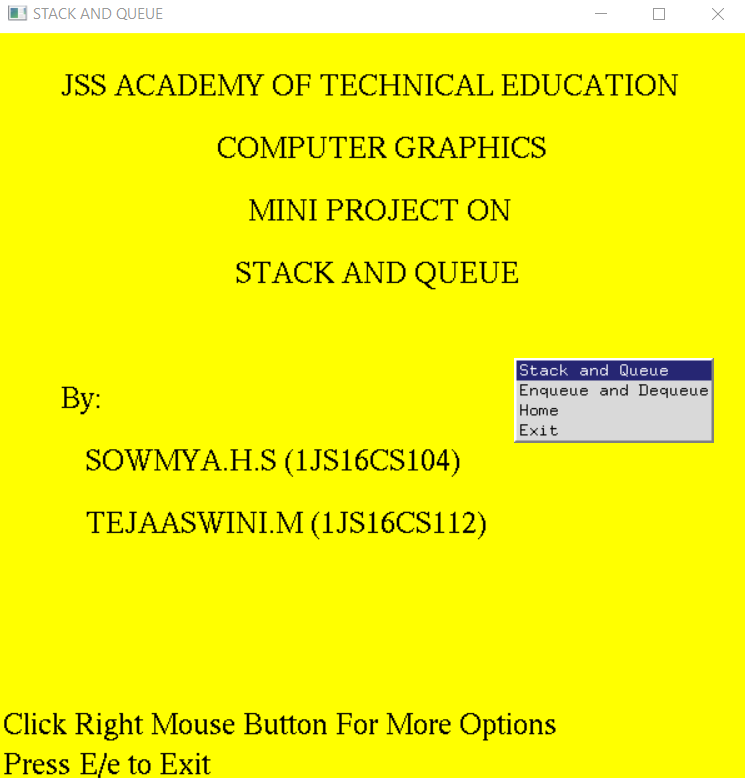
**Chapter 5**

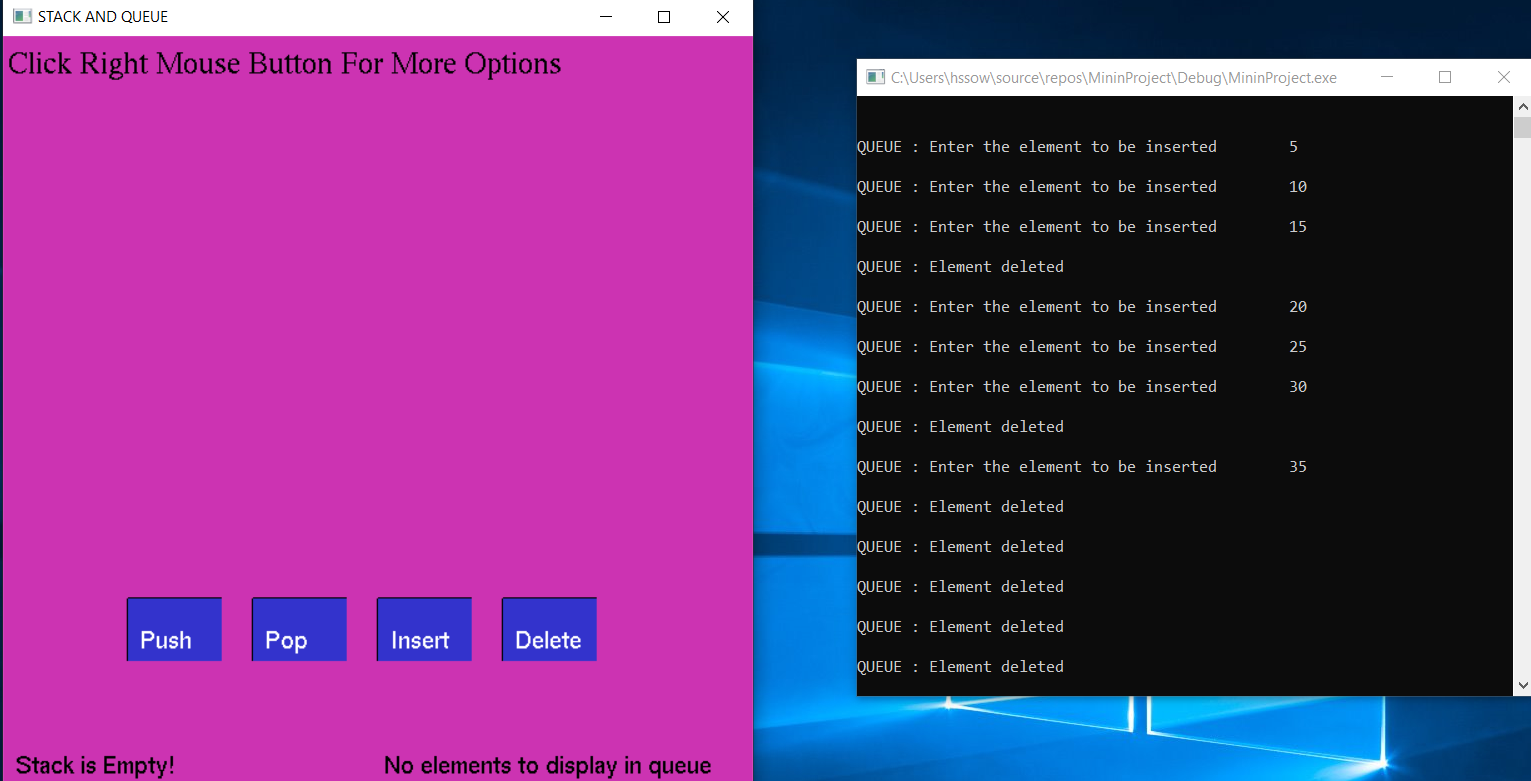
**Results and Discussions**

The project is executed using C language. We have put in few screen shots to show the working of Stack and Queue.

**Figure 5.1:** Home Screen

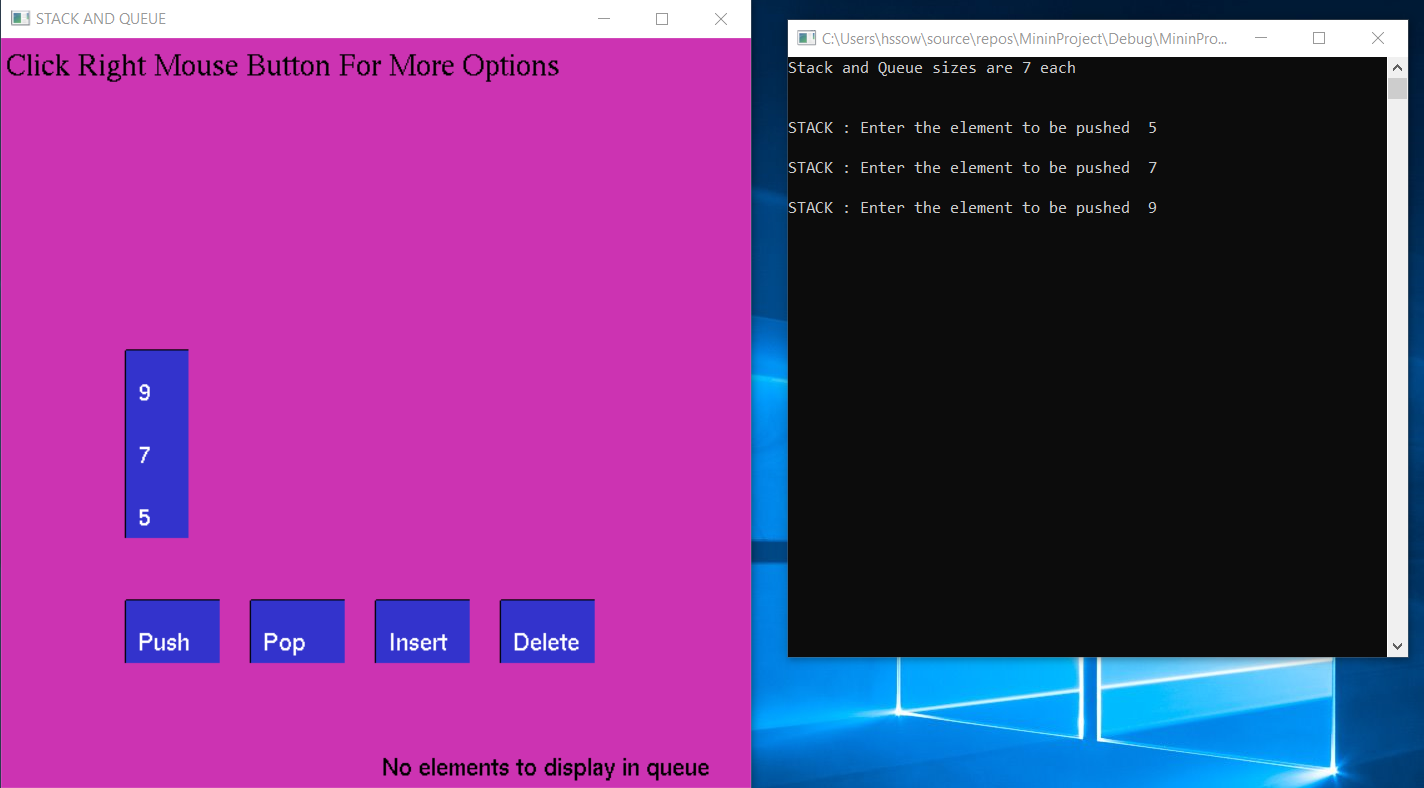
Figure 5.1 shows the home page of the program. It contains information about the title of the project, identities of the students participating in it. By pressing right click on mouse for more options you will get the menu. By pressing E/e key from keyboard you can exit.

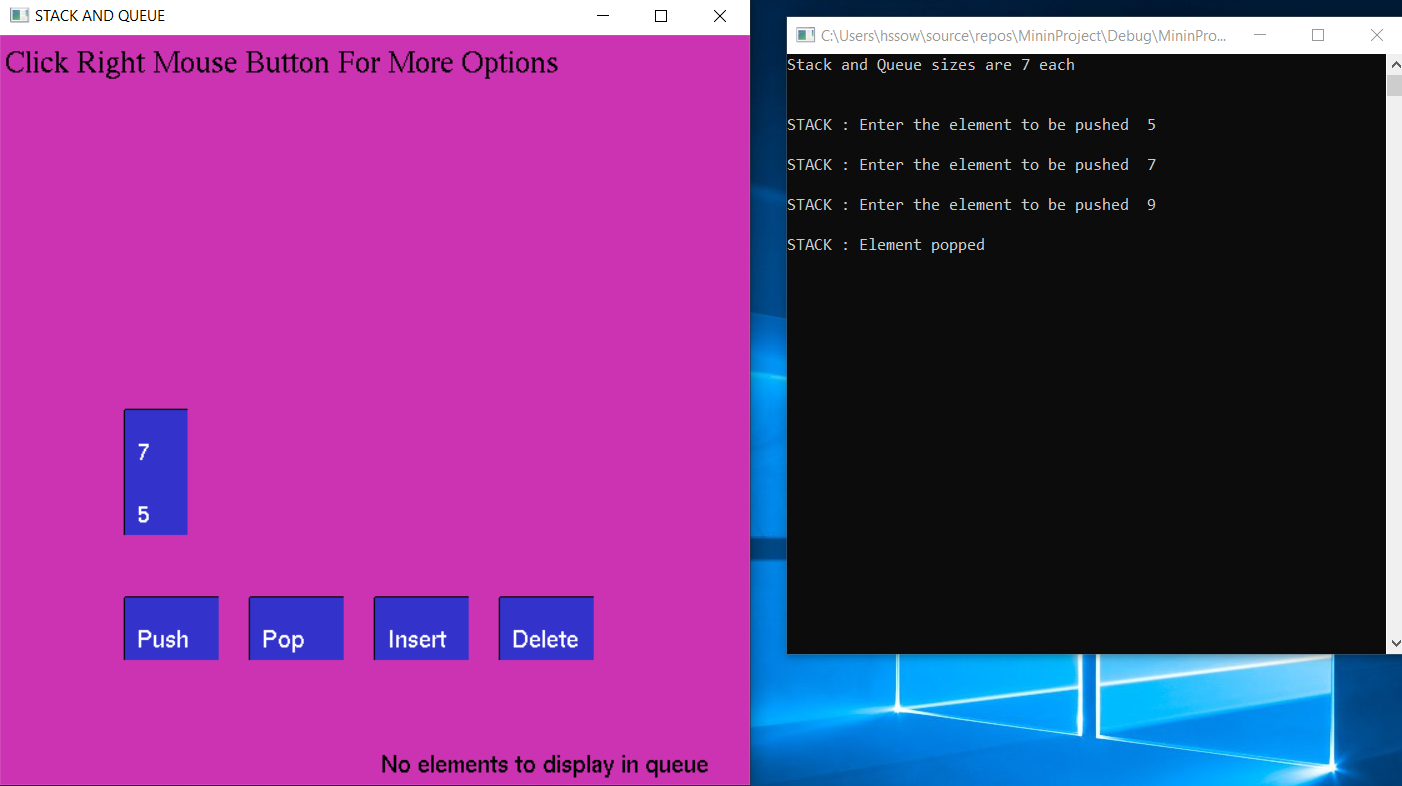
**Figure 5.2:** Main Menu

Figure 5.2 shows the main menu of the program. From here the user can navigate to various sections of the program by selecting one of the available options.

**Figure 5.3:** Visual Representation of the Window

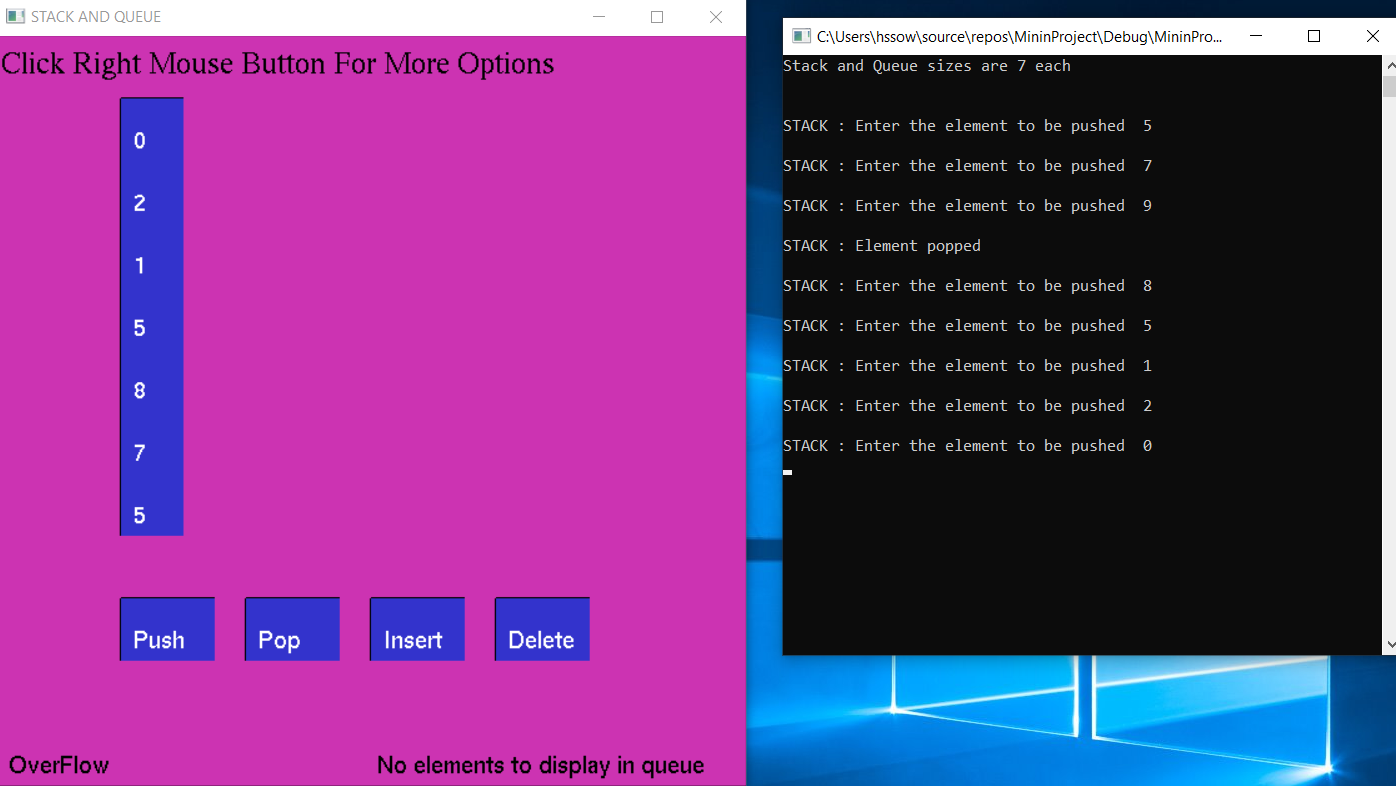
Figure 5.3 gives a visual representation of Push, Pop, Insert, and Delete. As there are no elements inserted for both Stack and Queue we can see the display message showing that the Stack is Empty! and No elements to display in queue.

**Figure 5.4:** Push Operation

Figure 5.4 gives a visual representation of the pushing the elements inside the stack. And user can input any integer from the keyboard.

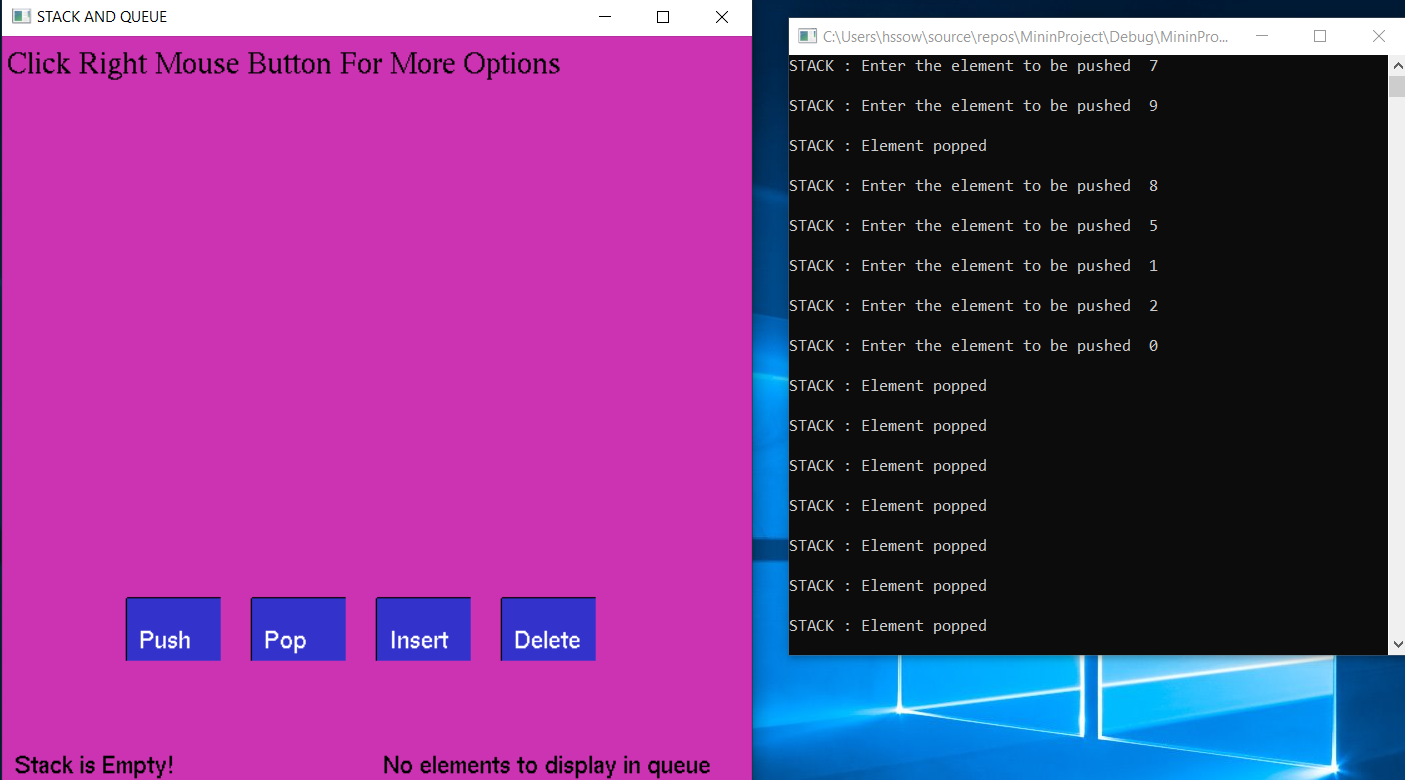
**Figure 5.5:** Pop Operation

Figure 5.5 shows that the top of the stack element is poped. Because stack follows FIFO i.e, first in first out.



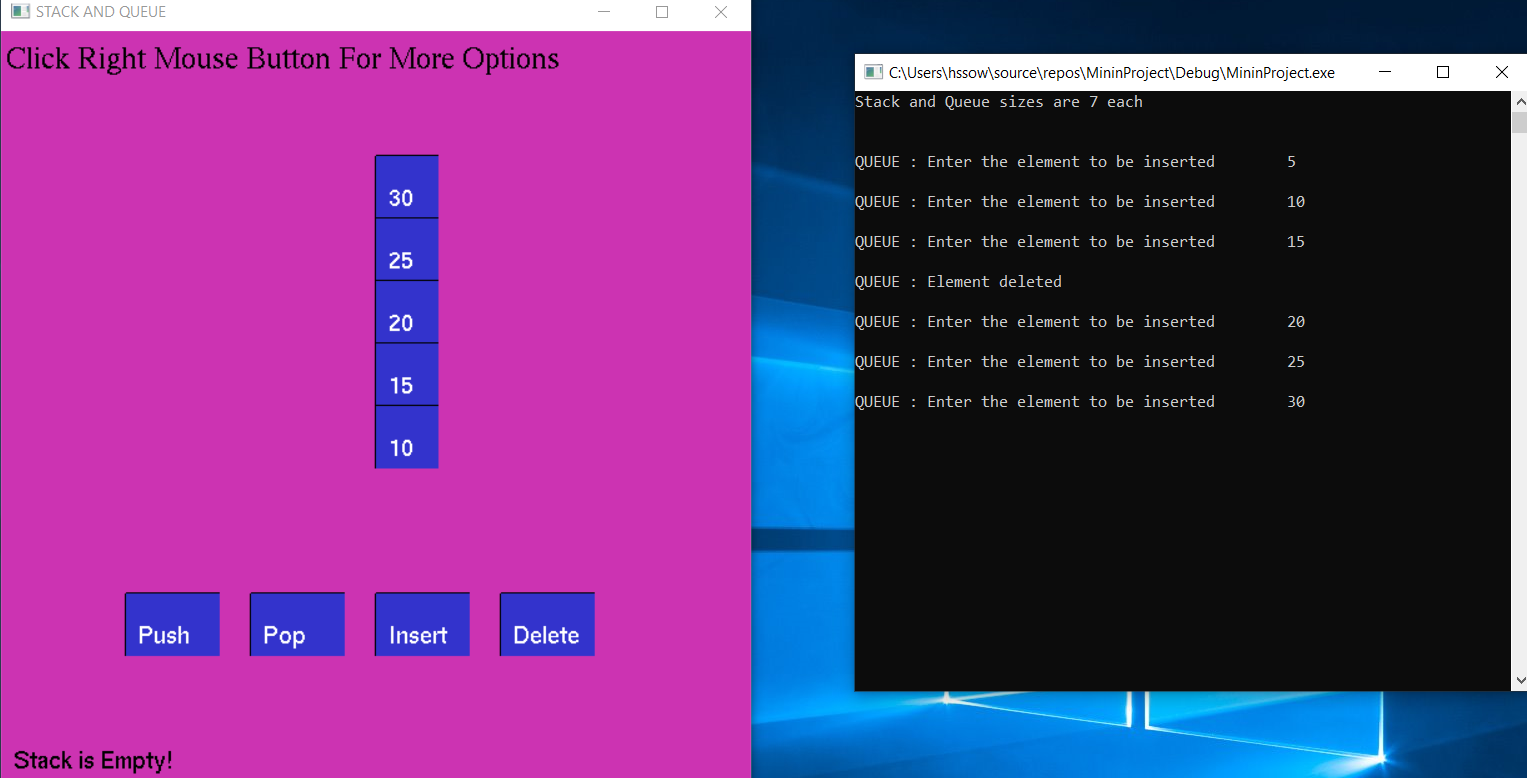
**Figure 5.6:** Stack Overflow

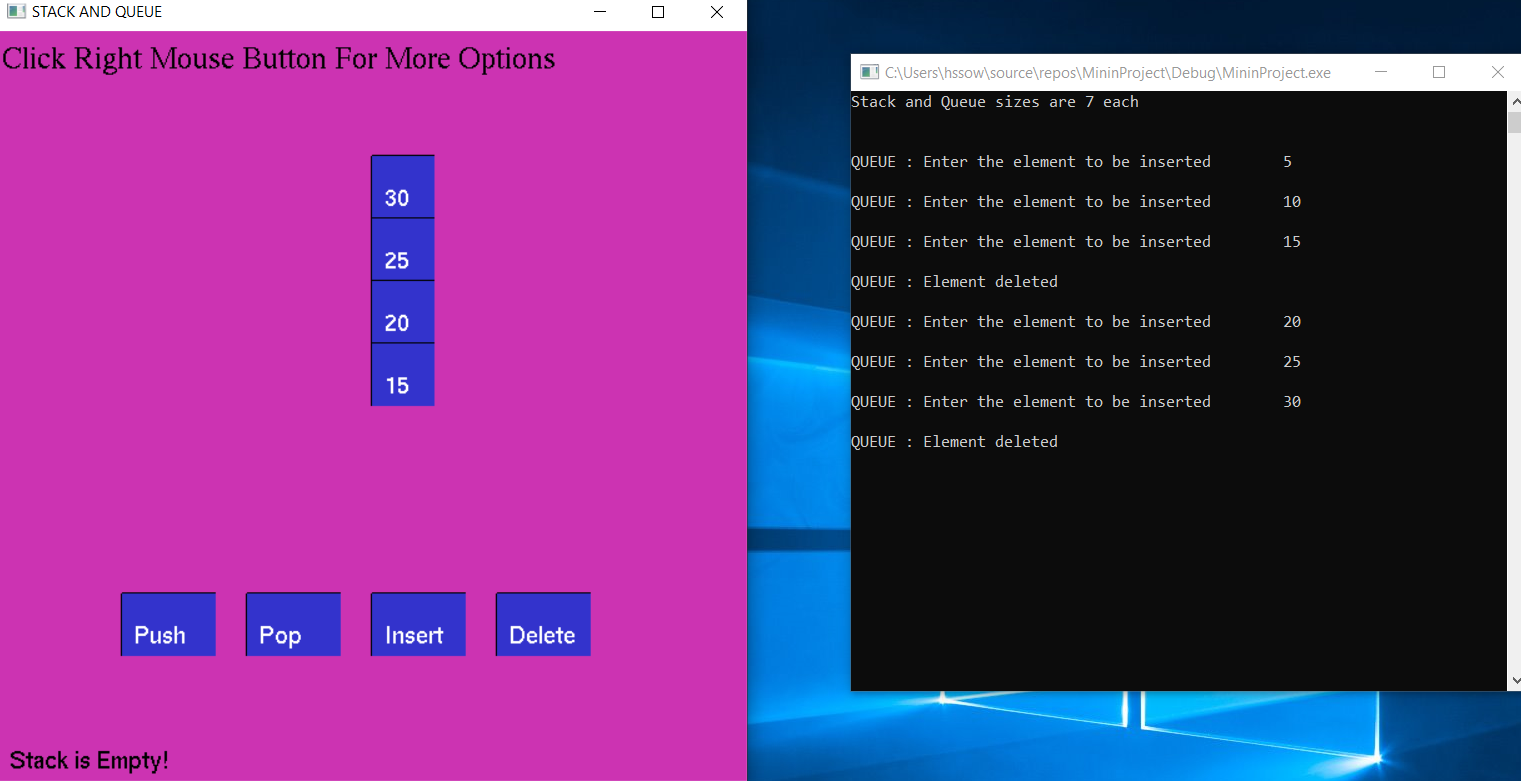
Figure 5.6 Once the elements exceed the array size then overflow occurs.



**Figure 5.7:** Stack Underflow

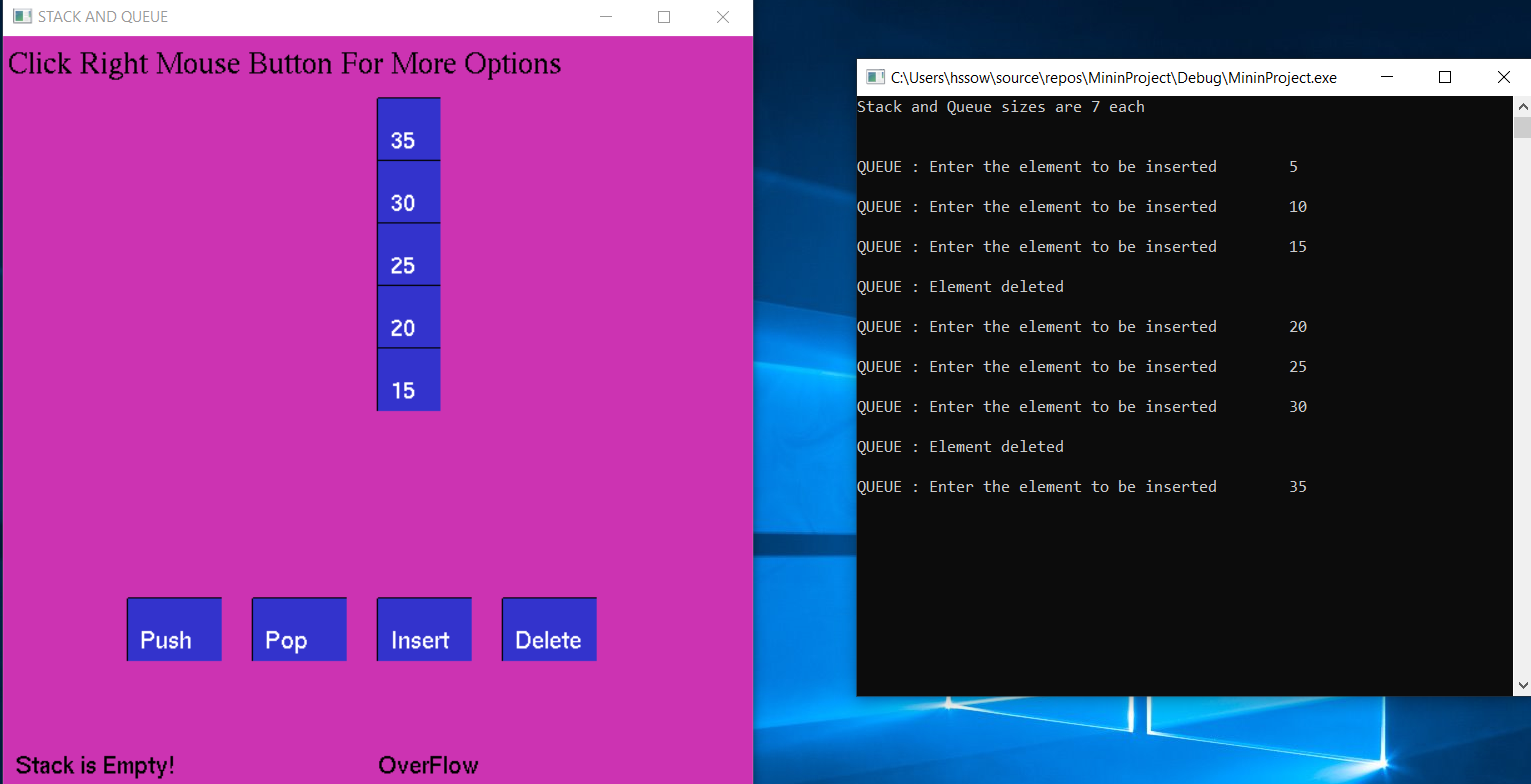
Figure 5.7 shows that all the elements are poped and when there is no elements in the stack then it is called is Underflow or Stack is Empty.

 **Figure 5.8:** Insert Operation (Queue)

Figure 5.8 shows that how elements gets inserted using Queue Operation.

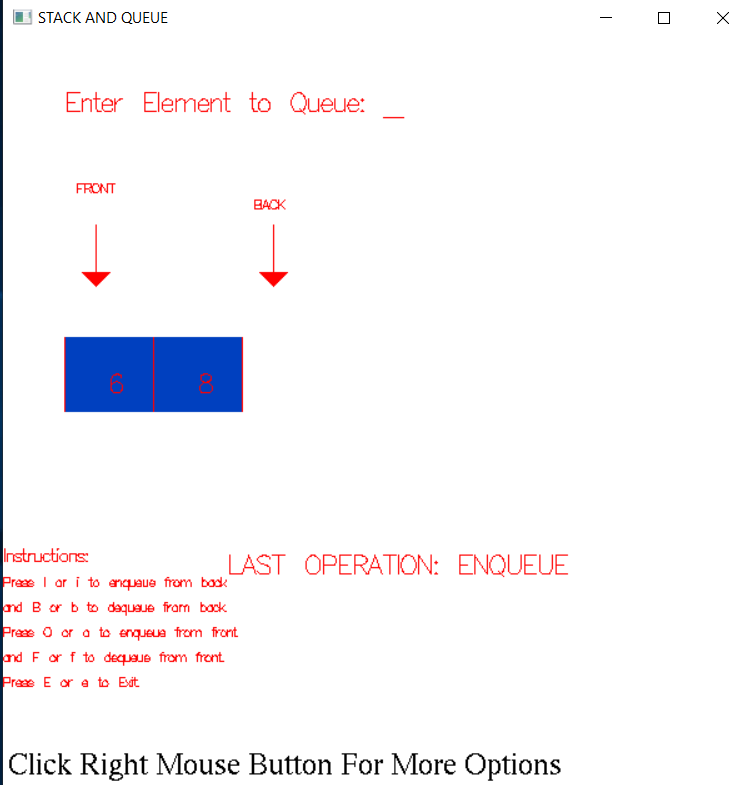
**Figure 5.9:** Delete Operation (Queue)

Figure 5.9 shows that how the element is deleted from the Queue. As it follows LIFO principle the element which was inserted at the end will be deleted first.



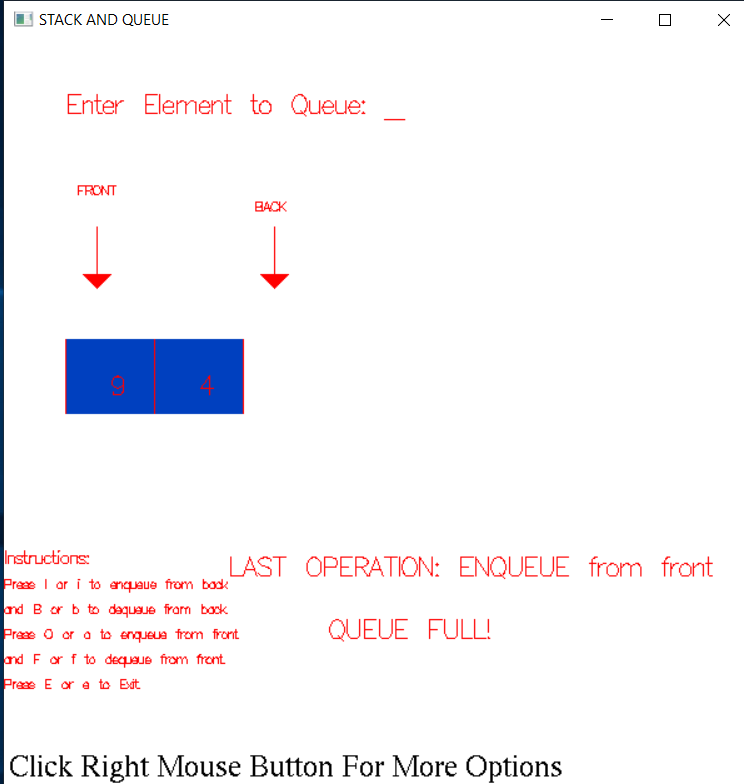
**Figure 5.10:** Queue Overflow

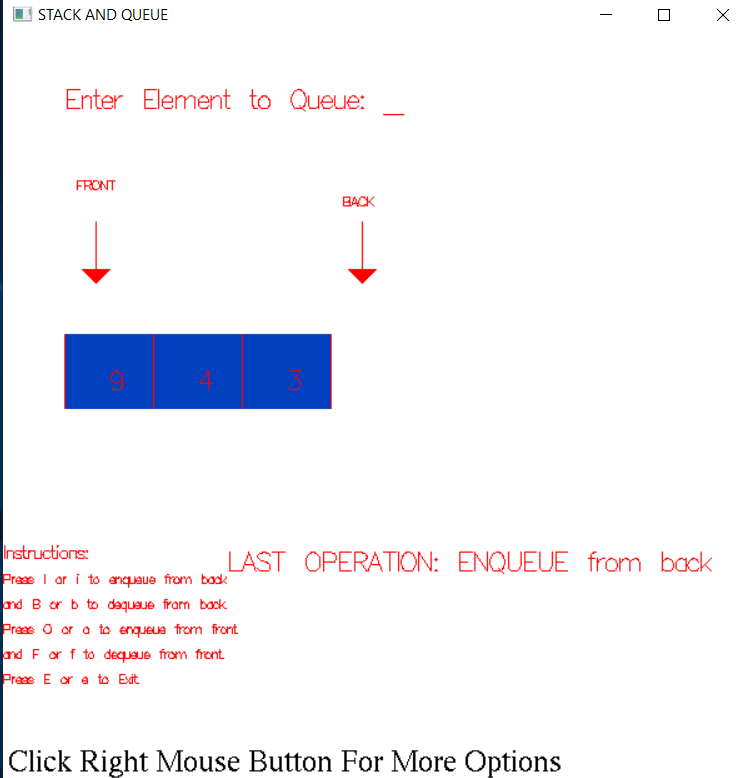
Figure 5.10 When the Queue is fully filled then and even when we insert elements Queue OverFlow Occurs.



**Figure 5.11:** Enqueue

Figure 5.11 Here we can add elements from both end of the Queue

 **Figure 5.12:** Enqueue from front

Figure 5.12 As we can see we cannot insert any element in front as it is already full

**Figure 5.13:** Enqueue from back

Figure 5.13 we can insert any element in from back end of the Queue.

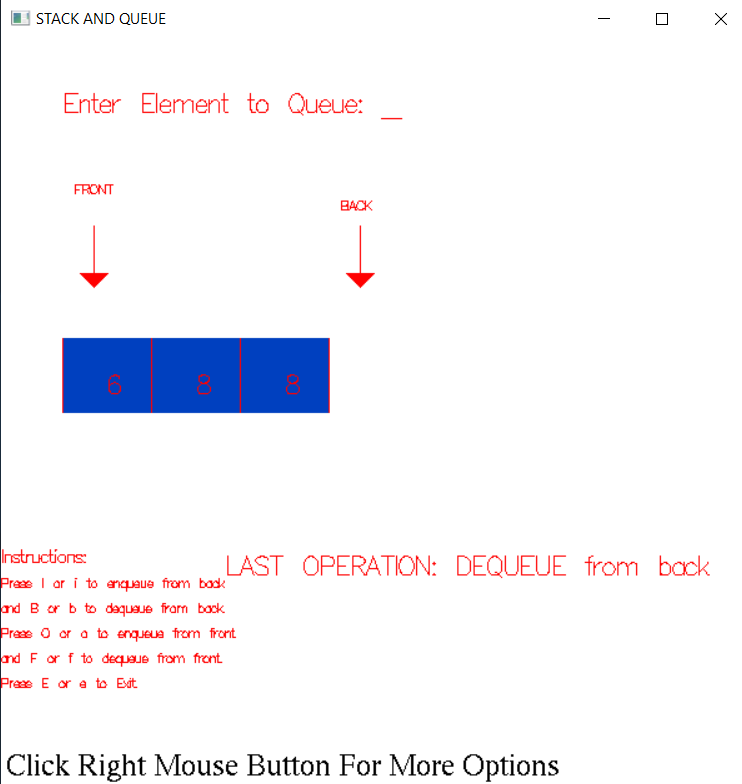
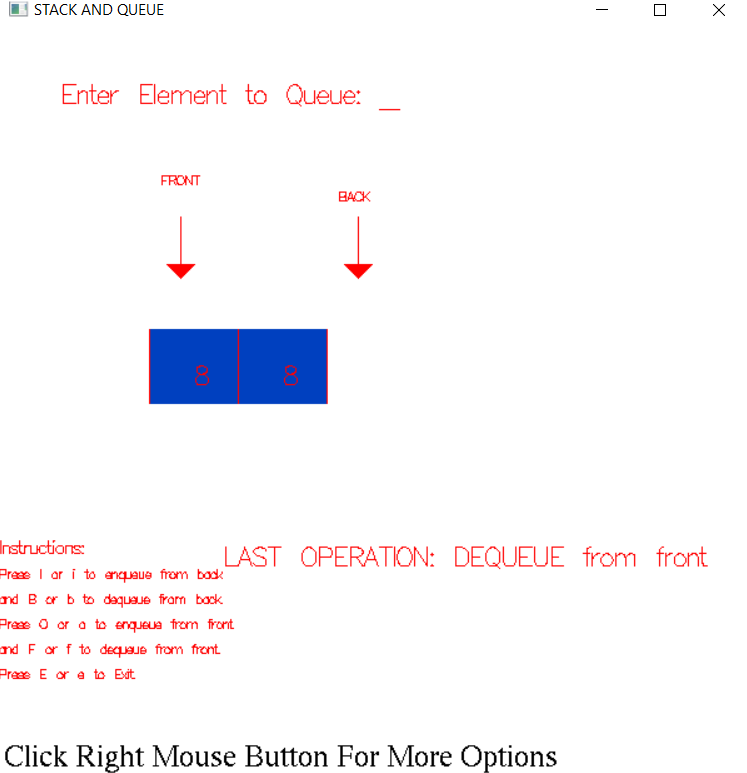
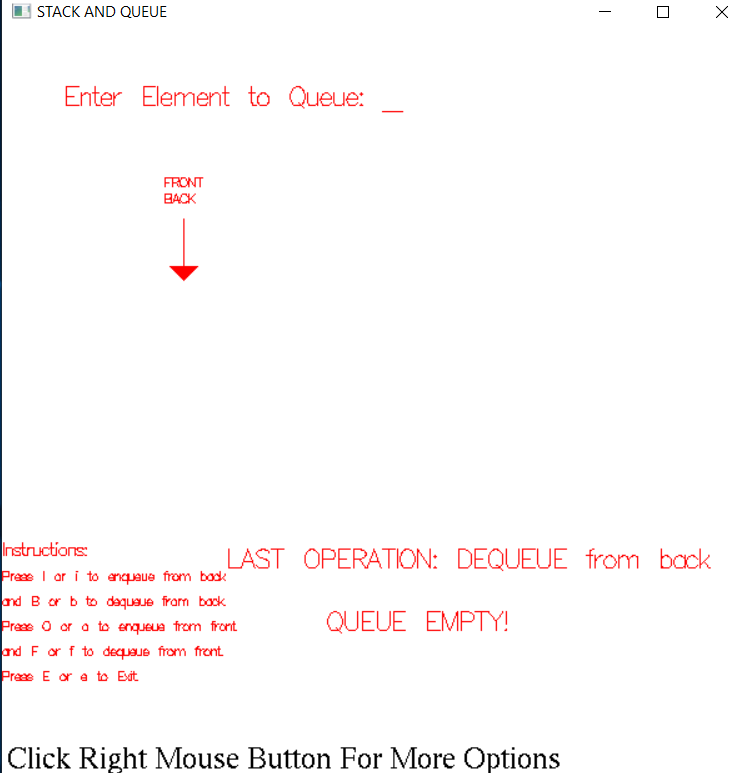
 **Figure 5.14:** Dequeue from back

Figure 5.14 Dequeue means we can delete the elements from both the end of the Queue. In this case we are deleting the element from the back.

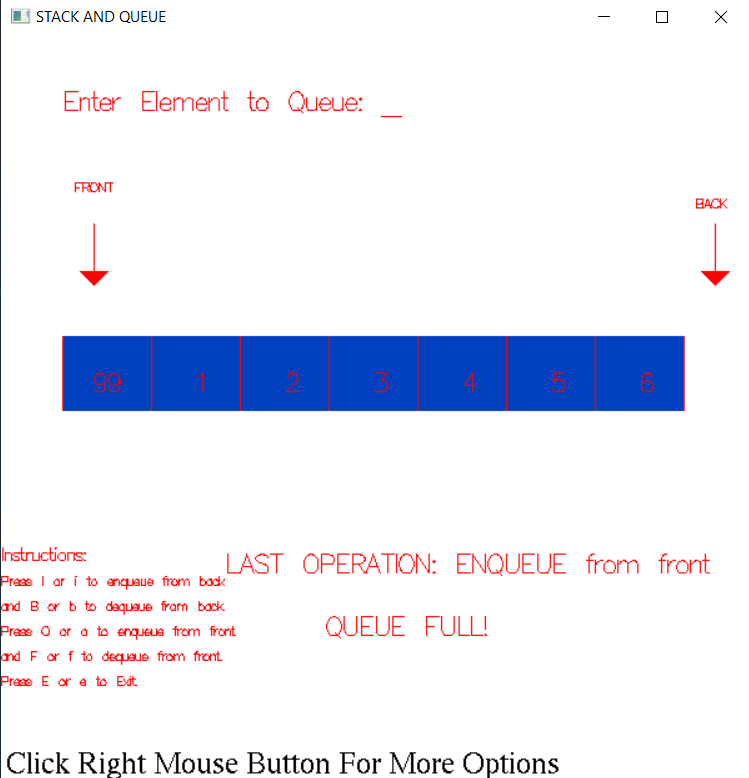


**Figure 5.15:** Dequeue from front

Figure 5.15 Dequeue means we can delete the elements from both the end of the Queue. In this case we are deleting the element from the front.



**Figure 5.16:** Queue Empty!

Figure 5.16 Shows an Empty Queue..

**Figure 5.17:** Queue is full!

Figure 5.17 It says that we cannot insert any elements as Queue is Full.

**Chapter 6**

**Conclusion and Future Enhancements**

**6.1 Conclusion**

This mini project on Stack and Queue using OpenGL is a reliable graphics package that provides the user with the basic working of the Insertion and Deletion. It provides a visual representation of Push, Pop, Insert, Delete, Enqueue and Dequeue. The user-friendly interface allows the user to interact with it very effectively.

**6.2 Future Enhancements**

The future scope of this project is vast. Using this project as base, similar gaming techniques can be implemented with very little modifications to the original code.

**References**

**Web Sources**

[1]:<https://en.wikipedia.org/wiki/Computer_graphics#Concepts_and_principles>

[2]:<https://en.wikipedia.org/wiki/OpenGL_Utility_Toolkit>

[3]: https://en.wikipedia.org/wiki/Stack\_(abstract\_data\_type)

[4]: https://www.tutorialspoint.com/data\_structures\_algorithms/dsa\_queue.htm

**Books**

[5] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd / 4th Edition, Pearson Education, 2011

[6] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL,5th edition. Pearson Education, 2008