**Chapter 1**

**INTRODUCTION**

This gives a brief introduction of the basic idea of Computer graphics softwares used in it.

* 1. **Computer Graphics**

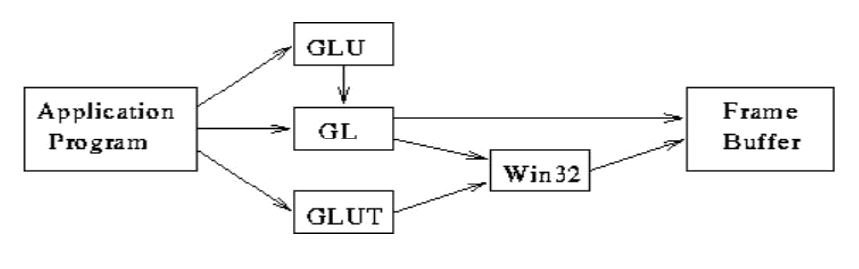
Computer graphics is an art of drawing pictures, lines, charts using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen[4]. Basically, there are two types of computer graphics namely. Interactive Computer Graphics involves a two-way communication between computer and user. Here the observer is given some control over the image by providing him with an input device for example the video game controller of the ping pong game. This helps him to signal his request to the computer. In non-interactive computer graphics otherwise known as passive computer graphics. it is the computer graphics in which user does not have any kind of control over the image. Image is merely the product of static stored program and will work according to the instructions given in the program linearly. The image is totally under the control of program instructions not under the user. Example: screen savers.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive[1]. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen.

**1.2. OpenGL Interface**

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. These offer functions to create and manipulate render lighting, coloring, viewing the models [3]. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects. Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are[6]:

* **Main GL:** Library has names that begin with the letter gl and are stored in a library usually referred to as GL.
* **OpenGL Utility Toolkit (GLUT):** provides minimum functionality that Should be accepted in any modern windowing system.

****

**Fig 1.2.1. Graphics pipeline**

Figure 1.2.1 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

**1.3. Project Description**

A **dequeue,** also known as a double-ended queue, is an ordered collection of items similar to the queue. It has two ends, a front and a rear, and the items remain positioned in the collection. What makes a deque different is the unrestrictive nature of adding and removing items. New items can be added at either the front or the rear[5]. Likewise, existing items can be removed from either end. In a sense, this hybrid linear structure provides all the capabilities of stacks and queues in a single data structure.

The deque operations are given below[7][10]

* Deque(): creates a new deque that is empty. It needs no parameters and returns an empty deque.
* add\_front(item) :adds a new item to the front of the deque. It needs the item and returns nothing.
* add\_rear(item) :adds a new item to the rear of the deque. It needs the item and returns nothing.
* remove\_front() :removes the front item from the deque. It needs no parameters and returns the item. The deque is modified.
* remove\_rear() :removes the rear item from the deque. It needs no parameters and returns the item. The deque is modified.
* is\_empty() :tests to see whether the deque is empty. It needs no parameters and returns a boolean value.
* size(): returns the number of items in the deque. It needs no parameters and returns an integer

# 

# 

# Chapter 2

**METHODOLOGY**

This section specifies the various functions used in the program and its implementation.

### Functions in OpenGL[6][8][9]

* + - **void glClear(glEnum mode);**

Clears buffers like color and depth.

### Void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);

### Sets the viewport.

### void glutSwapBuffers();

Swaps the front and back buffers.

### void glMatrixMode(GLenum mode);

Specifies which matrix will be affected by subsequent transformations

### void glLoadIdentity( );

Sets the current transformation matrix to identity matrix.

### void glutTimerFunc(unsigned int msecs, void (\*func) (int value), value);

### Registers timer callback to be triggered in a specific number of milliseconds.

### void glutBitmapCharacter(void \*font, int character);

It renders the character in the named bitmap font.

### void glRasterPos2i(GLfloat x, GLfloat y);

It is used to position pixel and bitmap operations.

### void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

Specifies the co-ordinates for clipping planes.

### void glutInit(int \*argc, char \*\*argv);

Initializes GLUT; the arguments from main are passed in and can be used by the application.

### void glutInitDisplayMode(unsigned int mode);

Requests a display with the properties in the mode; the value of mode is determined by the logical OR of options including the color model (GLUT\_RGB, GLUT\_INDEX) and buffering (GLUT\_SINGLE, GLUT\_DOUBLE).

### void glutCreateWindow(char \*title);

Creates a window on display; the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

### void glutMainLoop();

Causes the program to enter an event-processing loop.

### void glClearColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);

Specifies the red,green,blue and alpha values used by glClear to clear the color buffers.Values specified by glClearColor lies in the range 0 to 1

### void glutPostRedisplay(void);

It essentially sets a flag so that on the next iteration of the mainloop, your registered display() function is called.

### 2.2 User Defined Functions

* + - **void drawpoly(float a, float b, float c, float d);**

This function is used to display the polygon rectangle.

### void reshape(int w, int h);

This function is used to change shape and size of application window when required.

### void mouse(int button, int state, int mousex, int mousey)

This function is used to perform operations by clicking the mouse buttons accordingly.

### void display();

This function is used to display the elements and contents specified by respective functions.

### void keyboard(unsigned char key, int x, int y);

This function is used to perform operations by pressing appropriate keys on the keyboard.

### void printtext(const char s[], float x, float y, int l);

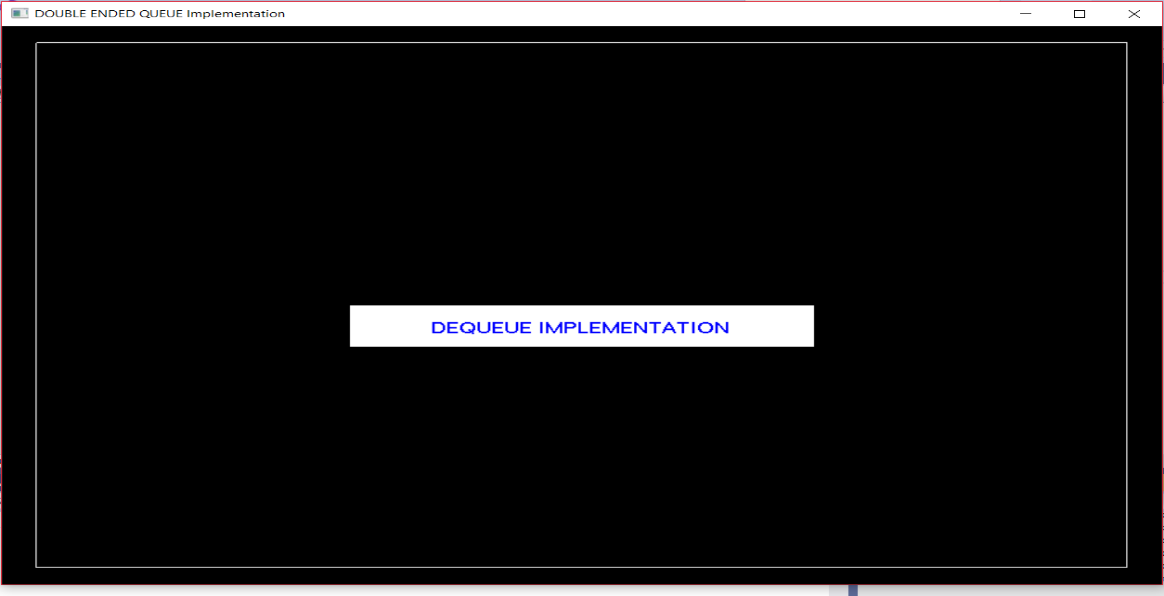
Prints the relevant texts where necessary.

# Chapter 3

**RESULTS**

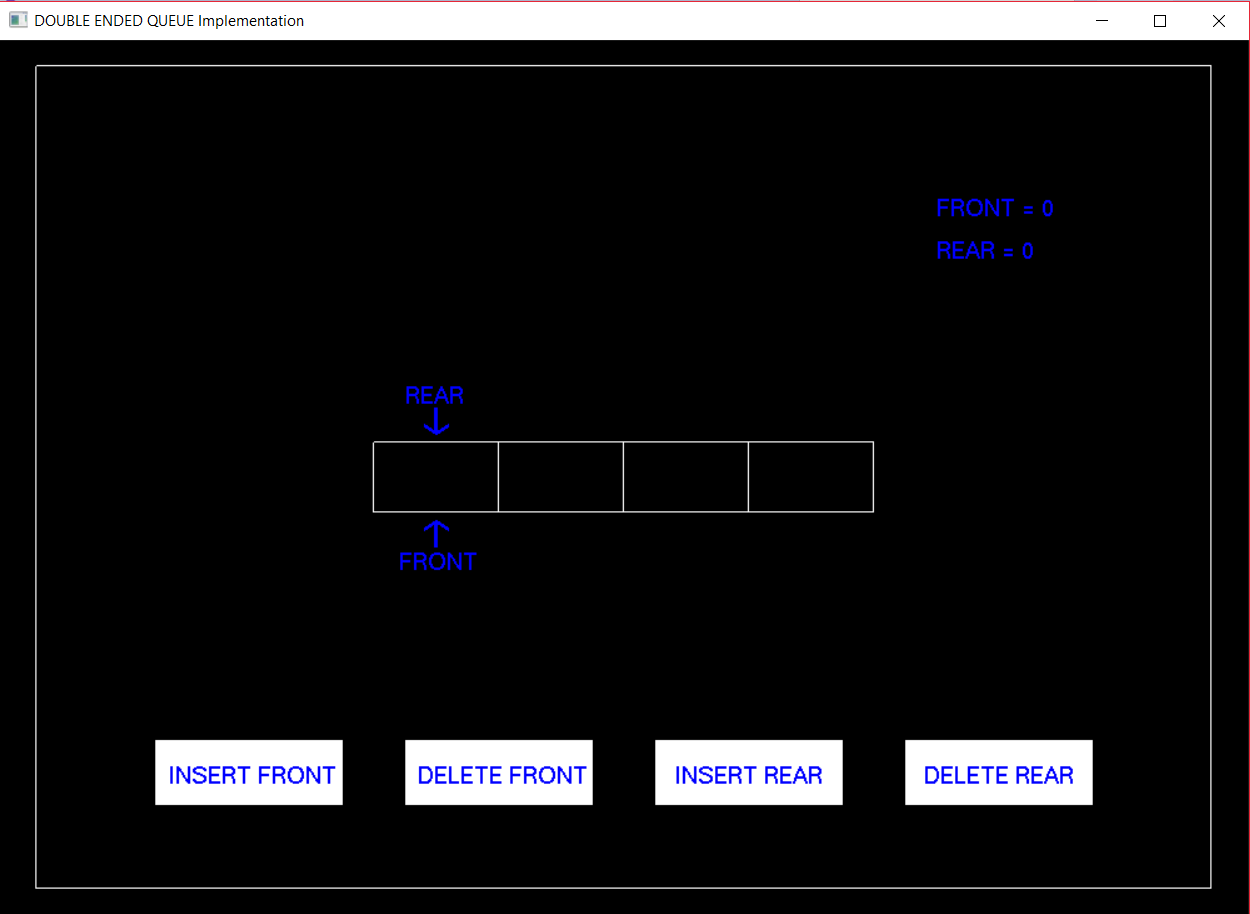
This section shows the graphical output of the program with relevant snapshots.

### Snapshots



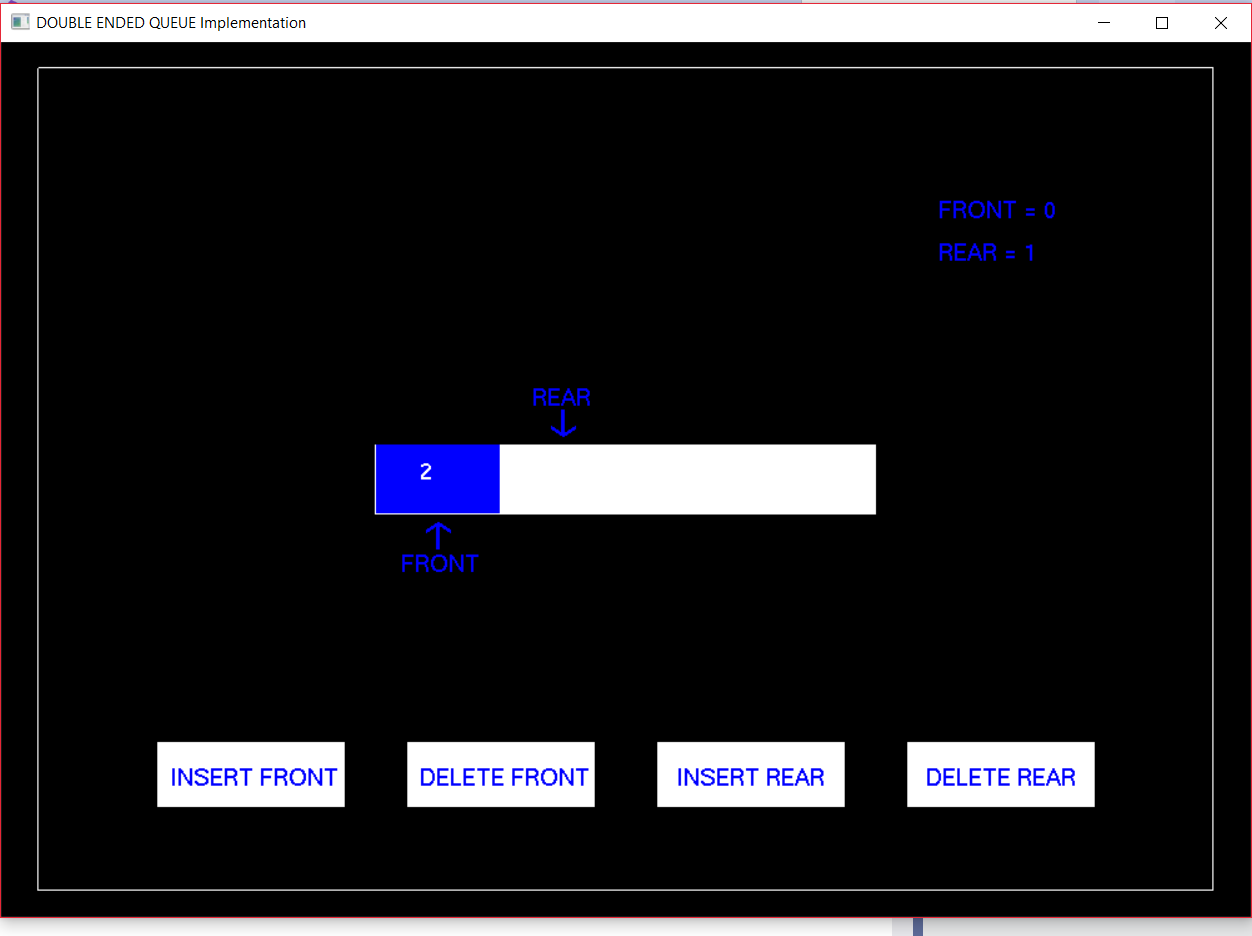
**Fig3.1.1: Starting screen of the output**

The above figure shows the first screen that appears after the queue size is given



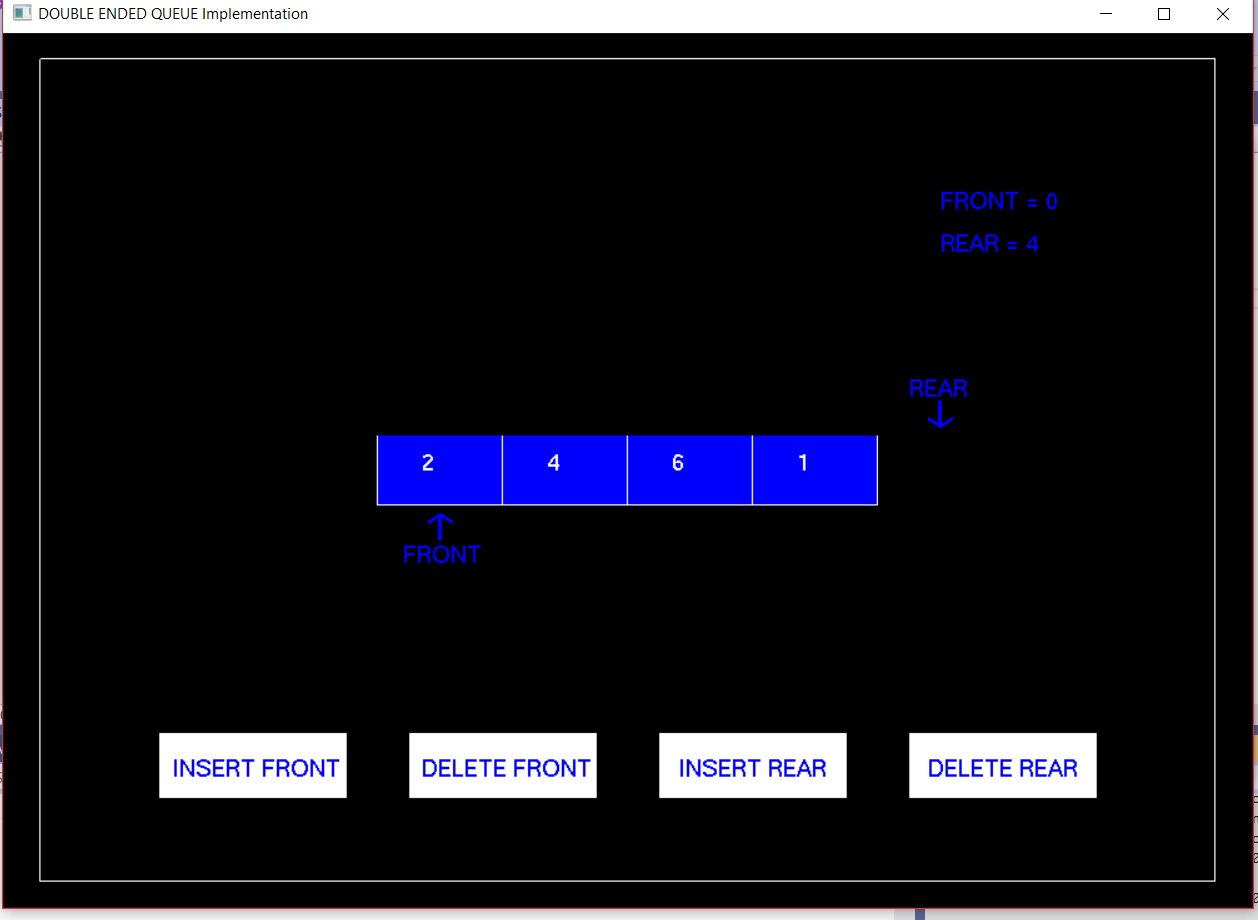
**Fig3.1.2: Selection of operation**

The above figure shows the window with all possible operations and the two pointers-FRONT and REAR.



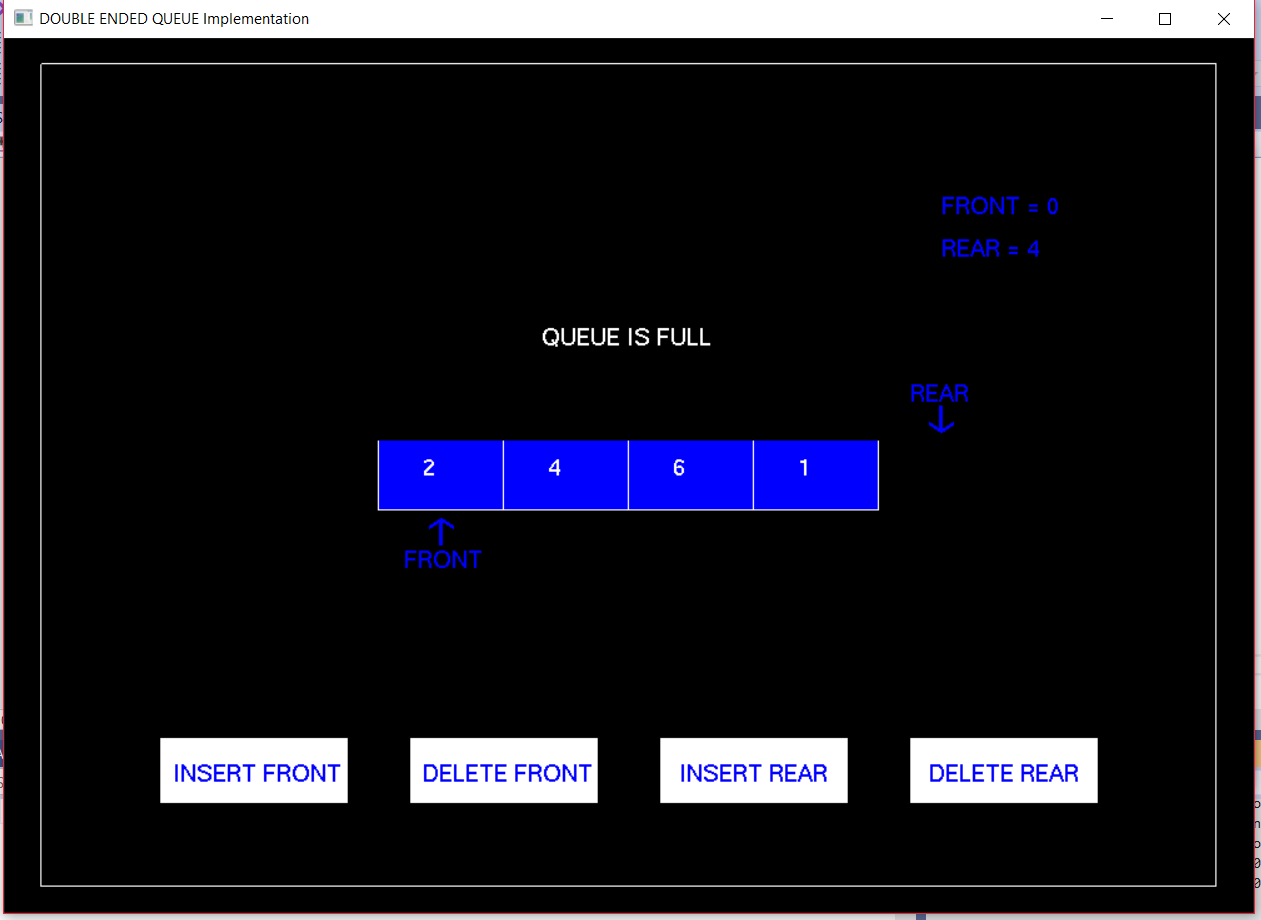
**Fig 3.1.3: Insertion from rear end**

The above snapshot shows how an element has been inserted at the rear end after selecting the option insert front from the base menu.



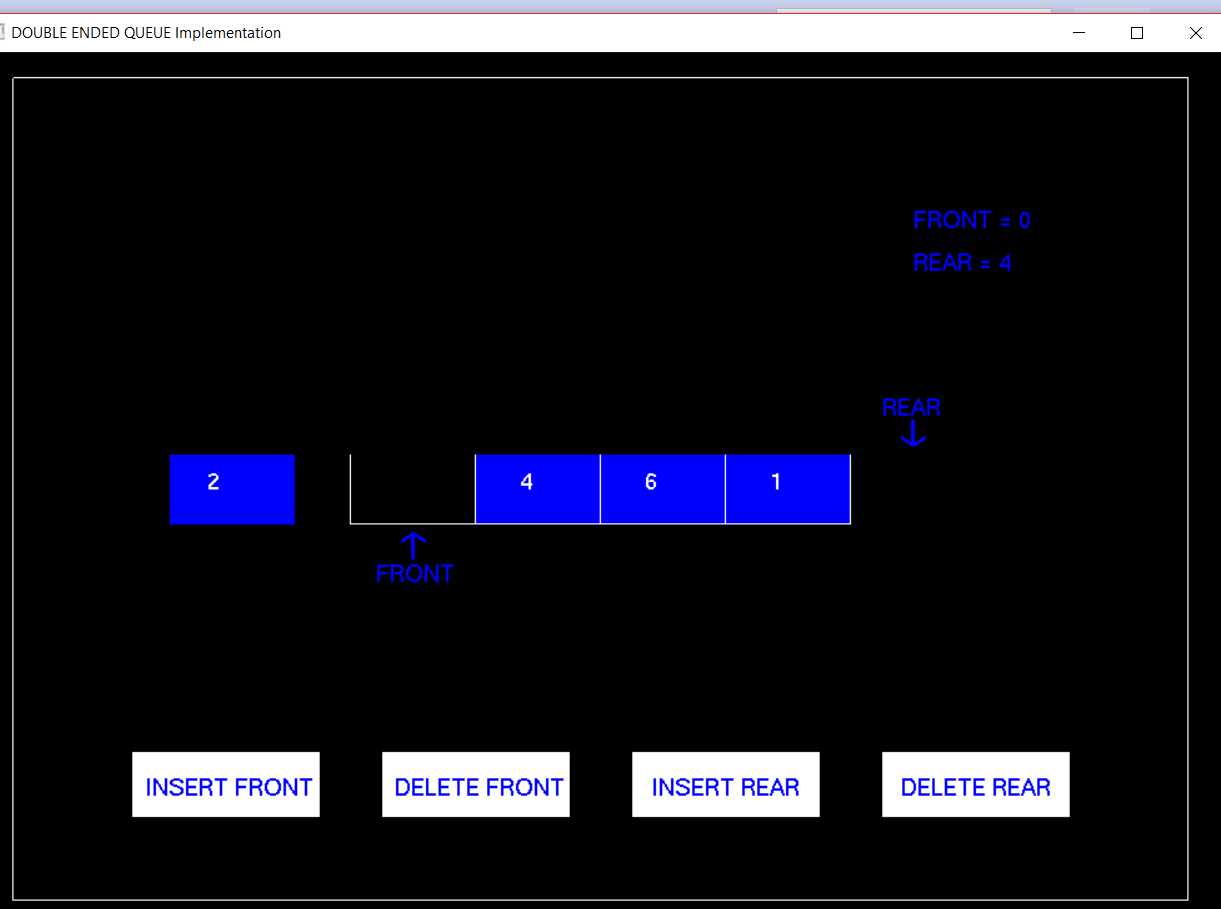
**Fig 3.1.4: Likewise, insertion of other three elements**

The above snapshot shows how all other elements have been inserted at the rear end after selecting the option insert front from the base menu.

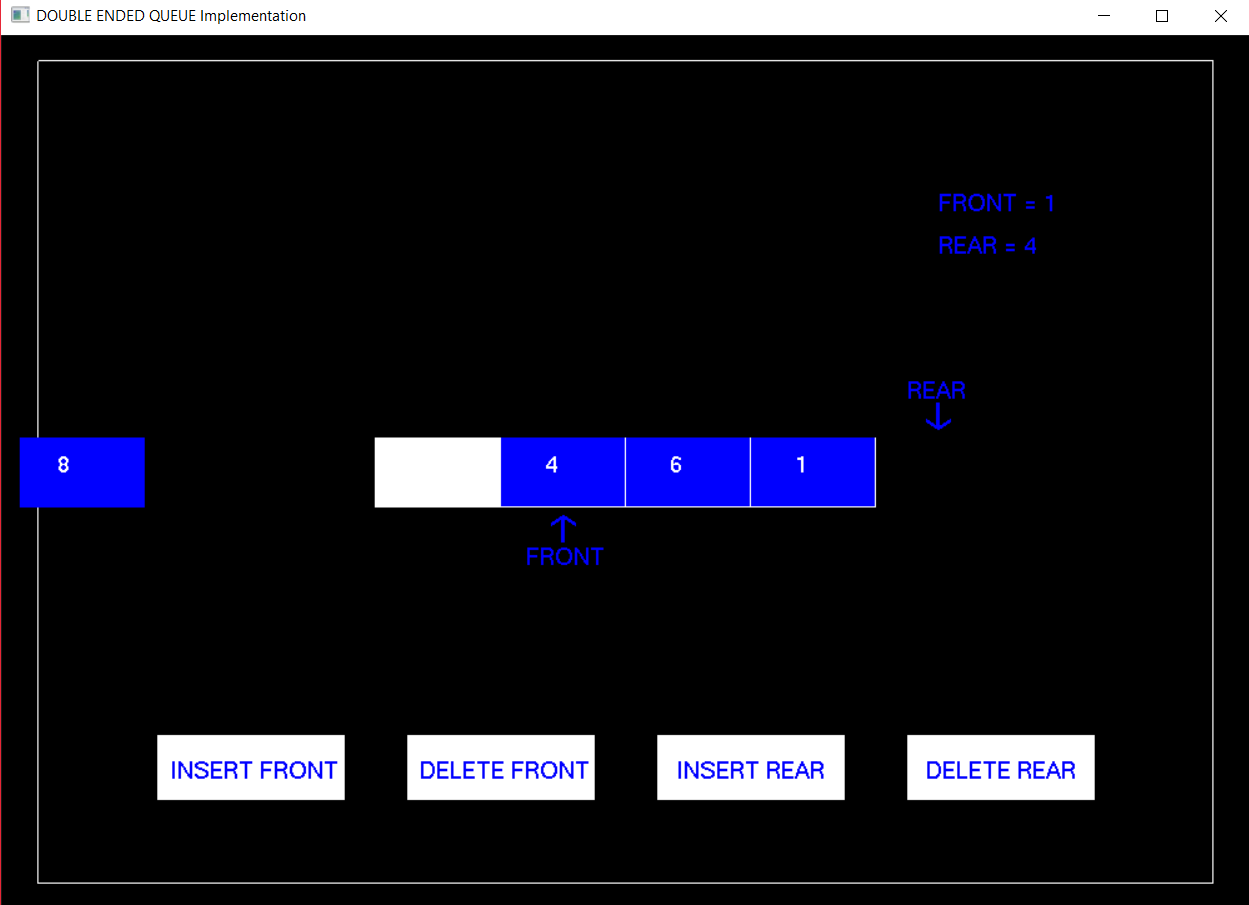


**Fig 3.1.5: Queue is full**

The above snapshot shows the condition when all elements have been inserted to the maximum and the queue is FULL

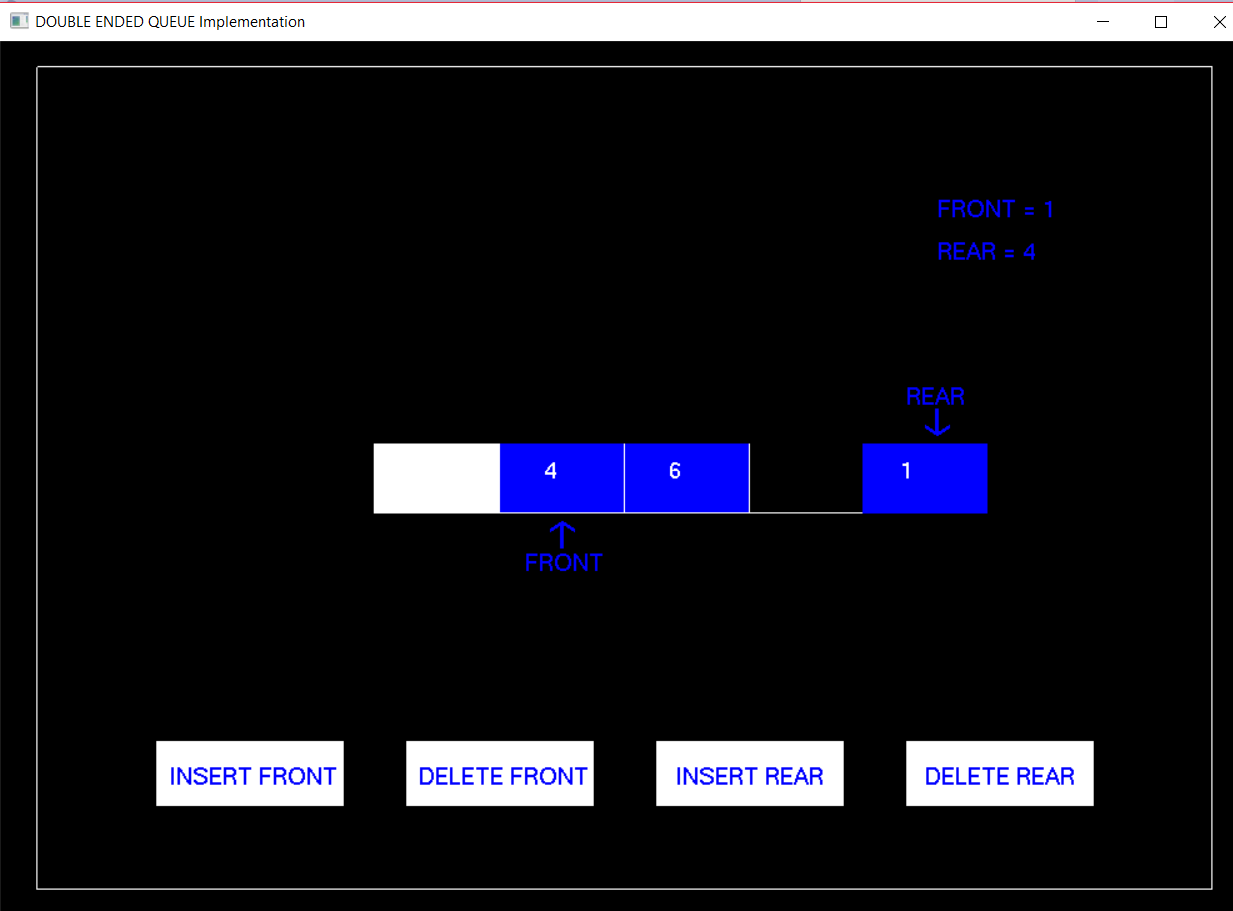


**Fig 3.1.6: Deletion from front end**

The above snapshot shows how an element has been deleted from the front end after selecting the option delete front from the base menu.

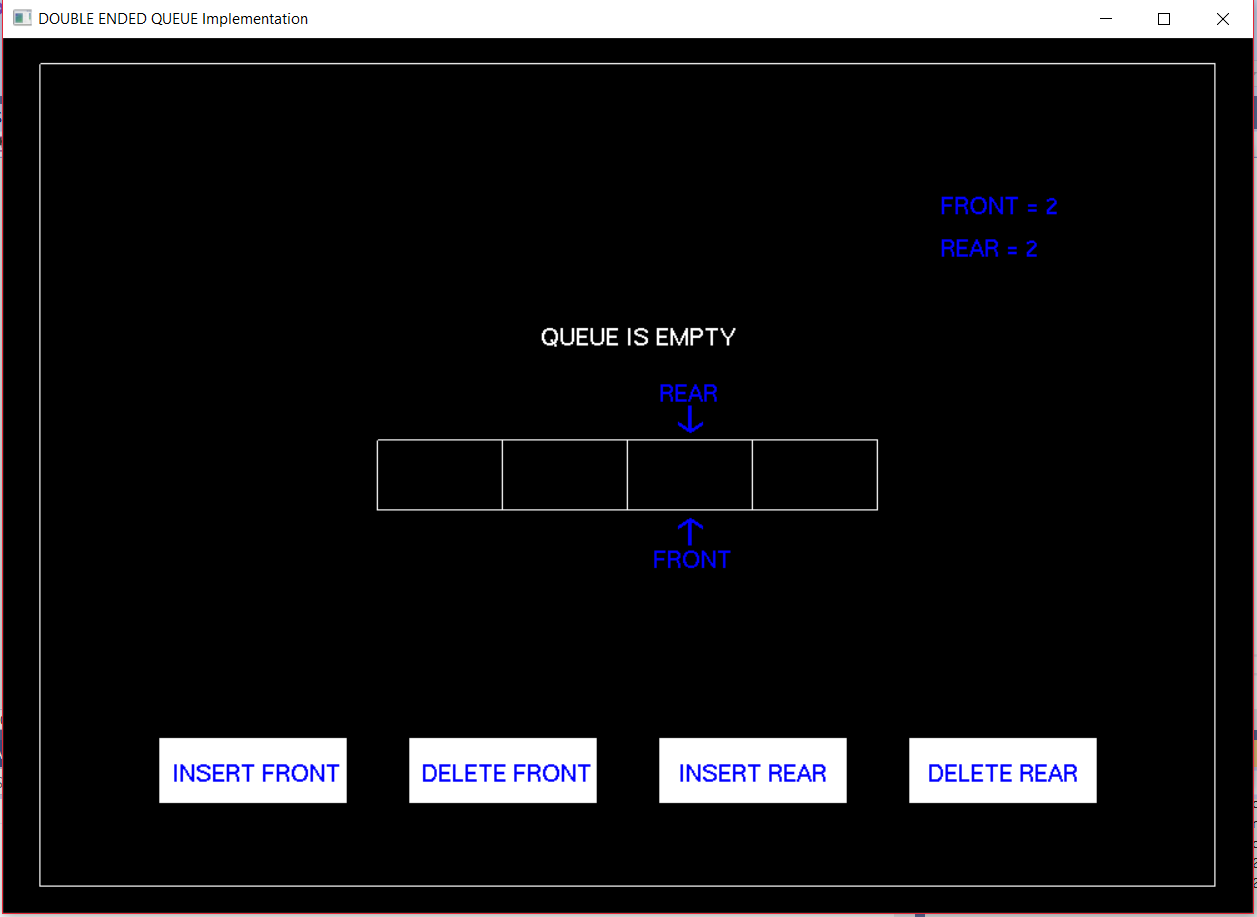
**Fig 3.1.7: Insertion from front end**

The above snapshot shows how an element has been inserted at the front end after selecting the option insert front from the base menu.



**Fig 3.1.8: Deletion from rear end**

The above snapshot shows how an element has been deleted from the rear end after selecting the option delete rear from the base menu.



**Fig 3.1.9: Queue is empty**

The above snapshot shows the condition when all elements have been deleted and the queue is EMPTY

# Chapter 4

**CONCLUSION**

The code we have implemented for our project is working well to the best of our knowledge. This project provided an opportunity to learn the various concepts of the subject in detail and provided us a platform to express our creativity. This project would be helpful in websites which hosts information related to data structures in a manner to help the students to learn about doubly linked list in a better way as it involves animation. This is a simple interactive application. It is user friendly and has features which makes it a simple graphics project. It is open source and has no security features. User is free to alter code for future enhancement.

# REFERENCES

* + 1. International Journal of Engineering Research Management Technology ISSN: 2348-4039 Page 45 September - 2014 Volume 1, Issue-5
    2. H. Sahni and A. Freed, Fundamentals of Data Structures in C 2nd edition(2014), Ch. 4, pp 190-195.
    3. OpenGL Superbible, 6th edition, Graham Sellers, Richard S. Wright and Nicholas Haemel, Addison Wesley, ISBN: 0321902947
    4. Allen Newell,Cliff Shaw and Herbert A. Simon"Computer Graphics"(2012), Ch 3, page 25-45
    5. Juan, Angel (2006). "Ch20 –Data Structures;ID06 - PROGRAMMING with OpenGL by CayS. Horstman" (PDF).p.3
    6. Woo,Neider, Davis, OpenGL Programming guide, MA,Reading: Addison-Wesley Developers Press, 2005
    7. "Definition of a double ended queue". National Institute of Standards and Technology. 2004-08-16.
    8. Woo, Neider, Davis, OpenGL Reference Manual, MA, Reading: Addison -Wesley Developers Press, 2003
    9. Schroeder,Martin,Lorensen, The visualization tool kit: An Object- Oriented approach to 3D graphics, New Jersey,Upper Sadle River:Prentice-Hall,Inc.,2001

[10] Antonakos, James L.; Mansfield, Kenneth C., Jr. (1999). The Practical Data

Structures Using C/C++. PrenticeHall. pp.165–190. ISBN 0- 13-280843