****

Data Structures and Algorithms

Semester 4 / 6 (2020/21)

**SCHOOL OF INFOCOMM TECHNOLOGY**

Diploma in Information Technology

Diploma in Cybersecurity and Forensics

**Assignment**

**Duration: 18** January to **7** February 2021 (Week 14 to Week 16)

**Weightage:** 40% of Module

**Individual/Team/Both:** Team of 2 Students (Team with Individual Component)

|  |  |
| --- | --- |
| **Student Name** | **Student Number** |
| 1. **Tan Yuan Ming** | **S10198319C** |
| 1. **Gladys Chua Ling Hui** | **S10196678G** |

|  |
| --- |
| ***WARNING***  ***If a student is found to have submitted work not done by him/her, he/she will not be awarded any marks for this assignment. Disciplinary action will also be taken.***  ***Similar action will be taken for the student who allows other student(s) to copy his/her work.*** |

[**Application Description**](#_foytbmla2d5j) **3**

[Introduction](#_7v7sj31h0f1s) 3

[Roles and contributions](#_t0iep5v1svg4) 3

[**Instructions**](#_oux7n02d3ik8) **3**

[**Data Structures and Algorithms Implemented**](#_phia177u99ys) **4**

[**Selected Data Structures & Algorithms**](#_dckpr88evbt4) **6**

[Benefits/Reasons](#_qk5gm52pi3im) 6

[Queue List](#_f0x83ripikgq) 6

[Array-Based List](#_its23fi6dqnm) 6

[Linked-List](#_gbwkcrudvs9k) 6

[List Searching (Sequential Search)](#_sy4a96aypxiw) 6

[Array List Sorting](#_sytddzgp6jl7) 7

[Auto Sort List (Selection Sort)](#_rcqene54dkxl) 7

[**Other Advanced Feature:**](#_yoi8hy2iohco) **7**

[(Non-Data Structures & Algorithms) GUI:](#_bg1zgefks81h) 7

[**Appendices**](#_x34bunl19wir) **9**

[**References**](#_4whdm2eayjgg) **22**

# Application Description

## Introduction

HospitalQ™ is a program designed for counter operations specifically queueing and registering for queues. HospitalQ™ can be used to view patients’ past medical records and add patients to the medical record database. The program focuses more on the queue and database (list) functionality which provide the staff with a comfortable and user-friendly GUI. It is also used by smaller clinics as compared to an actual hospital counter.

## **Roles and contributions**

|  |  |
| --- | --- |
| **Yuan Ming** | **Gladys** |

|  |  |
| --- | --- |
| **Queue List** | **Array-Based List** |
| **List Searching** | |
| **Selection Sort** | |
| **Classes** | |
| **Self Sorting Linked-List** | **Graphical User Interface** |

# Instructions

1. Download the program files using 7-Zip File Manager (.7z).
2. Run the program using graphical or command interface.
3. Select option 1-9 depending on your desired choice as stated in the menu.
4. Option 0 will exit the program and any other strings or number not stated will prompt you to try again.
5. Option 1 must first be selected to use other options to populate the list of patients.

# Data Structures and Algorithms Implemented

* **Queue**
  + **Enqueue Patient from List to Queue**
  + **Queue Number Count in Node**
  + **Display Front Patient**
  + **Reset Queue Number**
  + **Dequeue Patient**
  + **Total number of Patient in Queue**
* **Array-Based List**
  + **Add Patient into List**
  + **Display Patient in List**
  + **Search Function**
  + **Sort Function**
* **Linked-List**
  + **Add Doctor Name and Assigned Patient in List**
  + **Display Doctor Name and Assigned Patient Information**
* **List Searching**
  + **Search for Patient using IC and display Patient’s medical records**
* **Selection Sort**
  + **Sort Medical Record List alphabetically when prompted**
* **Classes**
  + **Patient.h , Patient.cpp, MedicalRecordList.h, MedicalRecordList.cpp, Queue.h, Queue.cpp**
* **Graphical User Interface (Boost)**
* **Auto-Sort Linked List (Selection Sort)**
  + **Auto Sort by Doctor’s Name when adding new item to the list**
  + **Auto Sort by Doctor’s Name before printing the list**

# Selected Data Structures & Algorithms

## Benefits/Reasons

### Queue List

Queue List is suitable as the algorithm uses a “first-come, first serve” basis which allows the front of the list to be dequeued efficiently. Queue List always adds items from the back, making the list acts similar to actual queues. Queue List does not have to allocate memory in advance and can be accessed very quickly. Queue List is flexible in terms of its ability to expand or shrink its maximum size when required.

### **Array-Based List**

Array-based List can be accessed through its index, meaning that it can be non-sequential. The Array-Based List, the average and worst case time complexity for accessing the list is O(1).

### Linked-List

As the doctors will be in-charge of many patients, the list will be required to scale indefinitely. Since, we will be expecting many entries existing for each patient, insertion and deletion will be much more simpler to perform as the node pointer will have to change rather than the whole data itself. In case of element deletion the time complexity for an array list is O(n) whereas for linked list is just O(1).

### List Searching (Sequential Search)

As compared to binary search , sequential search is faster as compared to binary when it comes to smaller lists. As our code is based on a system used by small clinics, not much data will be in the lists as compared to hospitals. Binary search is also used on sorted array lists while sequential search is able to be used in all types which makes it more easier to use than binary search. When a key element matches the first element in the list, the sequential search algorithm is the best case because the time complexity is 0(n), where n is the number of elements in our list.

### Array List Sorting

As our list is considered small, selection sort is more suitable for our list. Selection sort is able to perform well as compared to other sorting algorithms for small lists. Selection sort is also an in-place algorithm and does not require any additional storage. It also relies on the placements of the elements in the list. If there are many that are not sorted, it will take a longer time, however, since ours is automatically sorted when we add patients and print the list, it is constantly being sorted and need not spend much time sorting as it may only need to sort a few elements at one time.

### Auto Sort List (Selection Sort)

As our list is considered small, selection sort is more suitable for our list. Selection sort is able to perform well as compared to other sorting algorithms for small lists. Selection sort is also an in-place algorithm and does not require any additional storage. It also relies on the placements of the elements in the list. If there are many that are not sorted, it will take a longer time, however, since ours is automatically sorted when we add patients and print the list, it is constantly being sorted and need not spend much time sorting as it may only need to sort a few elements at one time.

# 

# Other Advanced Feature:

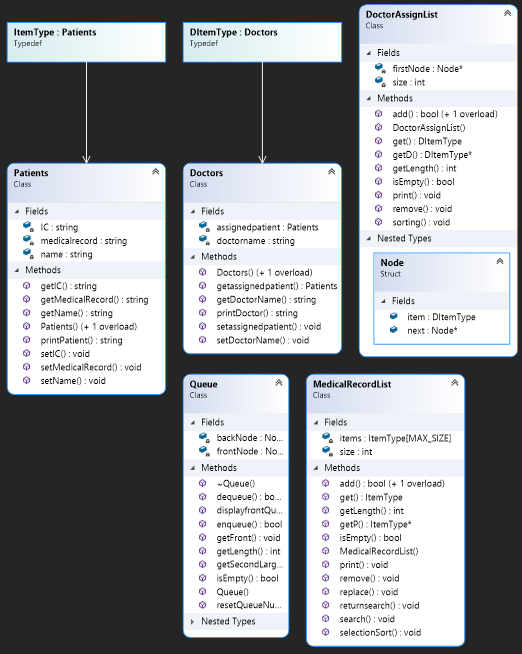
## (Non-Data Structures & Algorithms) GUI:

Juce was chosen as the GUI as it provided a more clean graphical interface as compared to a hard coded console menu. Juce as tools where it is able to display an interface with multiple images, colours and able to accept button commands. By using GUI, instead of using the command line interface, we are able to code it in such a way that it will contain buttons for checkboxes for us to select options rather than inputting texts to select. Thereafter, should the program require user input due to creation of doctor elements, patient elements, etc for classes, the program is to display text boxes and take in input accordingly, similar to the program without the GUI. However, due to time constraints and unsuitable labels and tools, we were not able to properly implement the GUI before the given time frame. As seen below in the appendices, are the attempts made on the GUI.

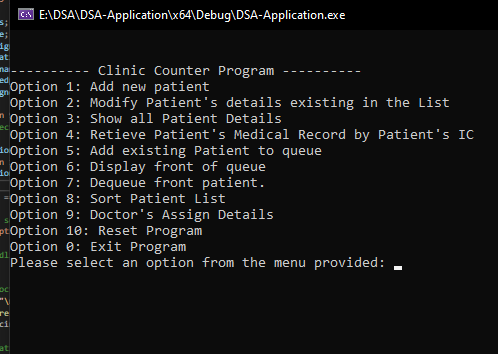
# 

# Appendices

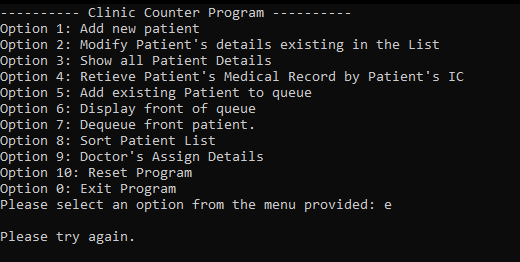
# 



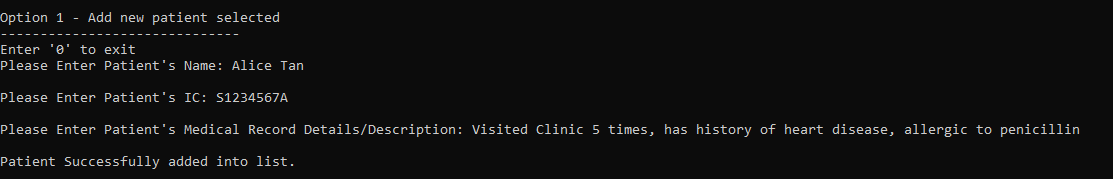
**Main Menu List:**

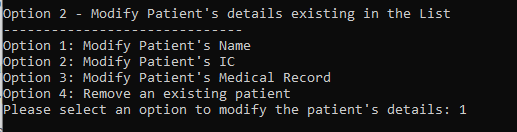


**Invalid Input:**

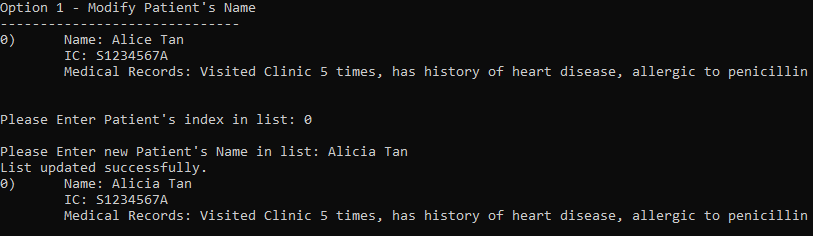
****

**Option 1: Add new Patient to list**

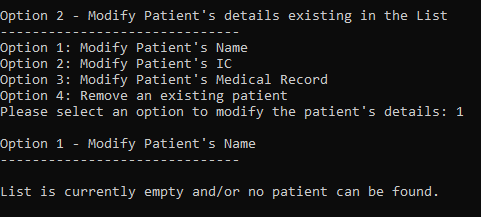


**Option 2 Main Menu: **

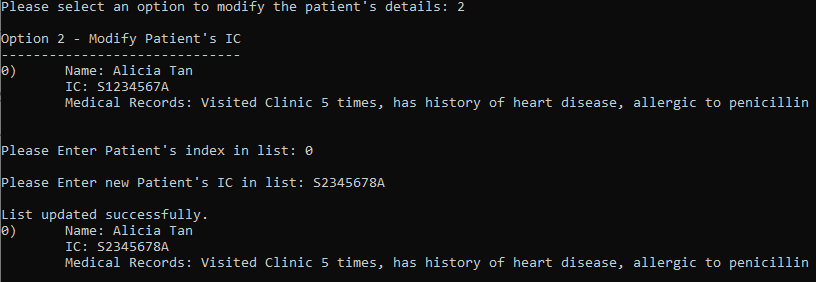
**Option 2 Modify Patient’s Name:**

****

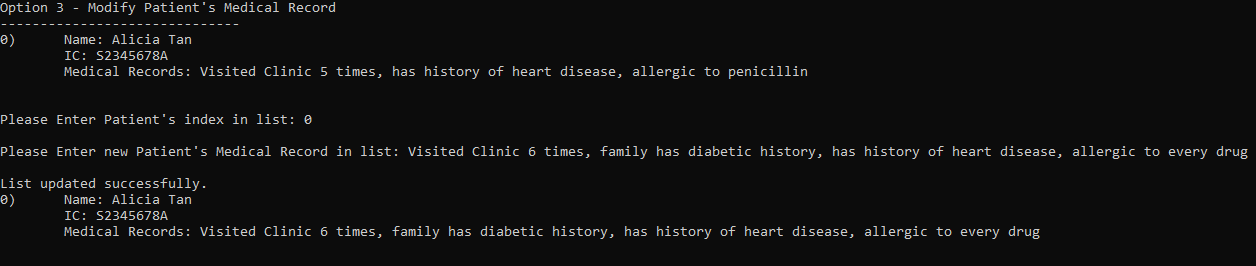
**No patient in list:**

****

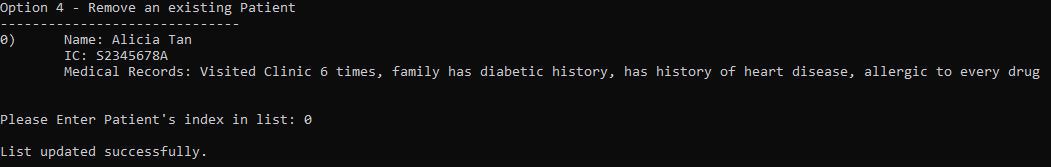
**Option 2 Modify Patient’s IC:**

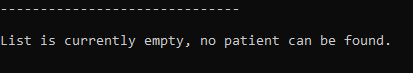
****

**Option 2 Modify Patient’s Medical Record:**

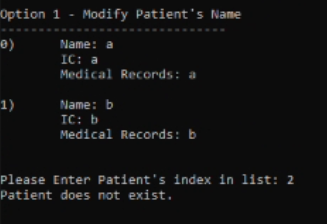
****

**Option 2 Remove an Existing Patient:**

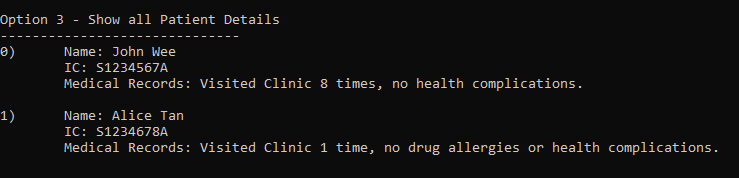
****

****

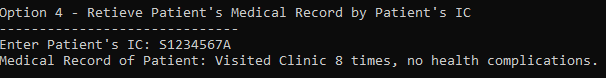
**If Patient Cannot be found:**

****

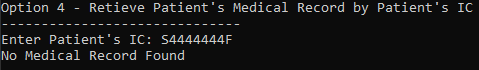
**Option 3 Show all Patient Details:**

****

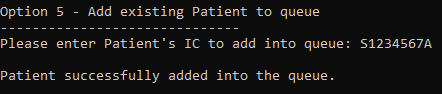
**Option 4 Retrieve Patient’s Medical Record by Patient’s IC:**

****

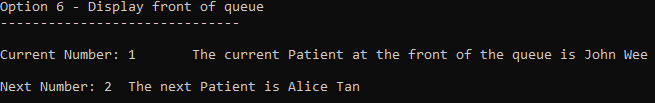
**No patient matching IC in List:**

****

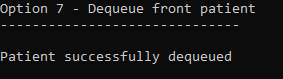
**Option 5 Add existing Patient to queue:**

****

**Option 6 Display front of queue:**

****

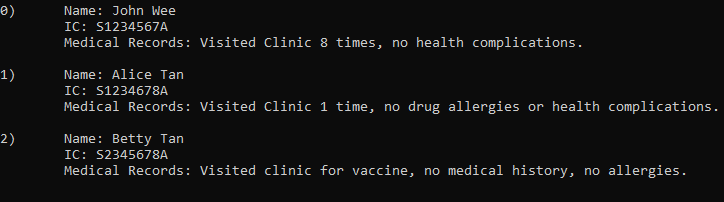
**Option 7 Dequeue front Patient:**

****

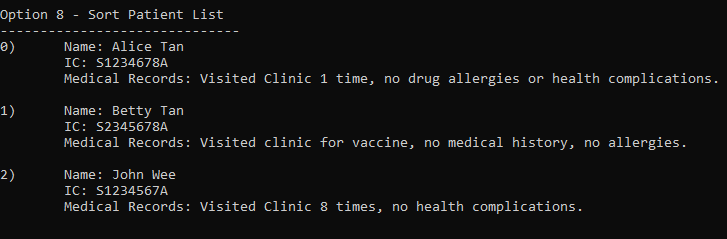
****

**Option 8 Sort Patient List:**

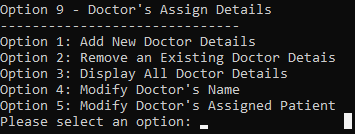
**Before:**

****

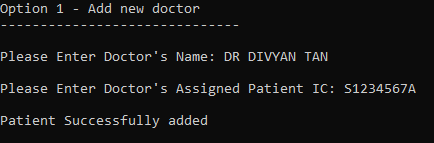
**After:**

****

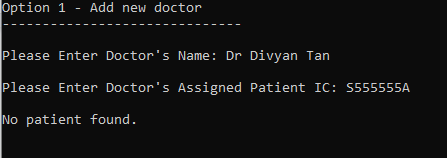
**Option 9 Doctor’s Assign Details Main Menu:**

****

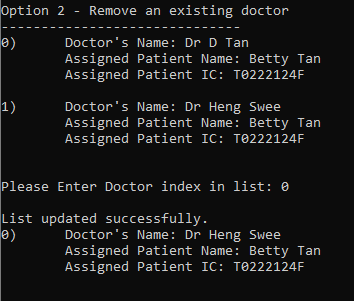
**Option 9 Add New Doctor Details:**

****

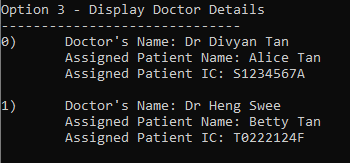
**Patient does not exist:**

****

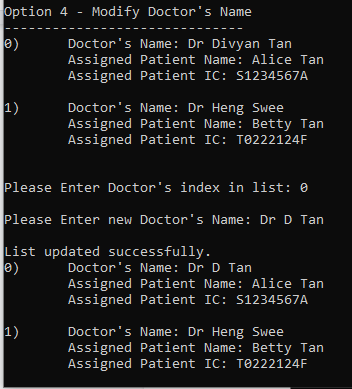
**Option 9 Remove an Existing Doctor Details:**

****

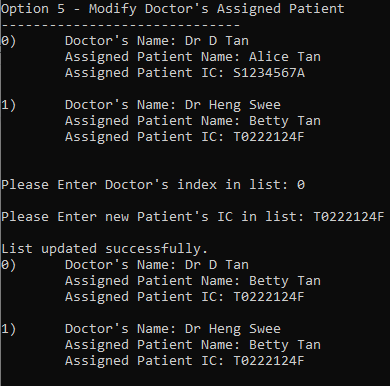
**Option 9 Display All Doctor Details:**

****

**Option 9 Modify Doctor’s Name:**

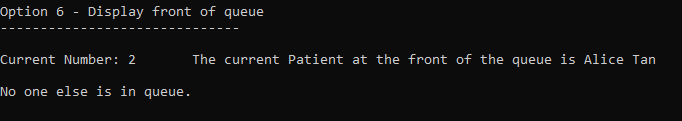
****

**Option 9**

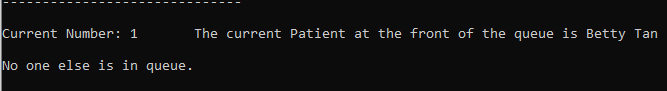
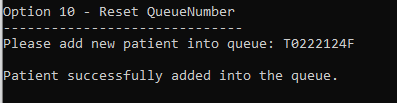
****

**Option 10: Reset Queue Number:**

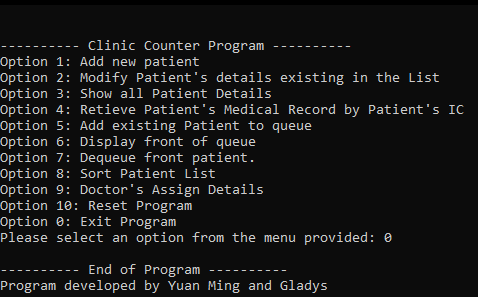
**Before:**

****

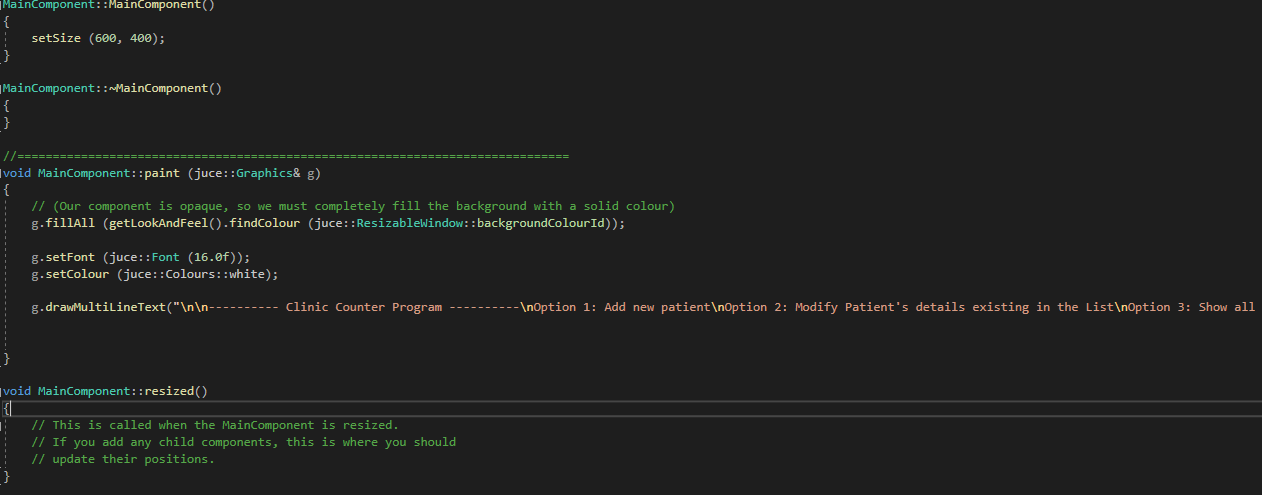
**After:**

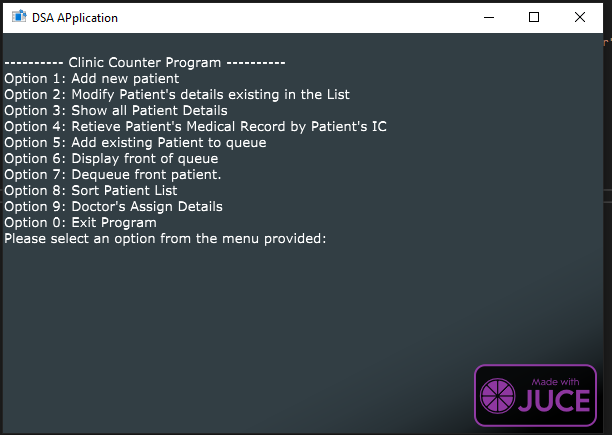
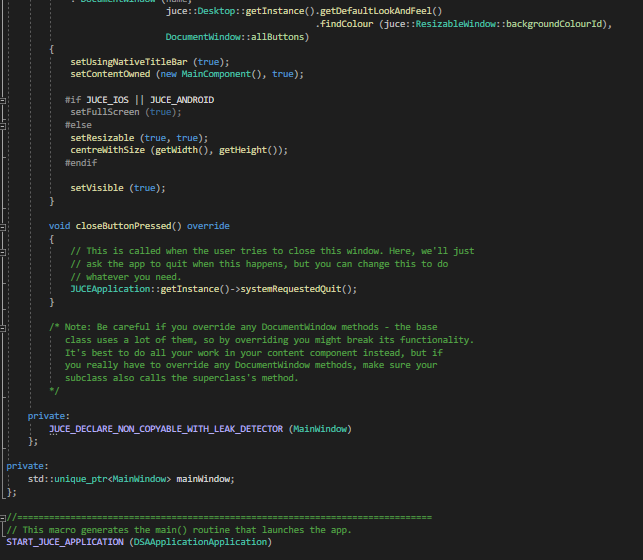
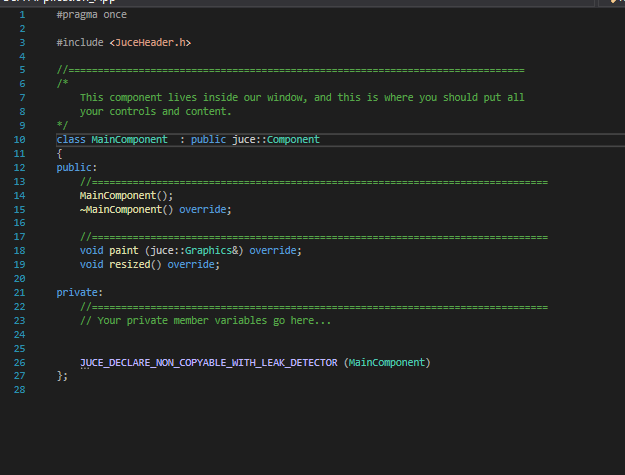
****

**Option 0: Exit Program:**



**Tried GUI:**





# 

# References

JUCE. “JUCE.” *JUCE*, 2021, <https://juce.com/> .Accessed 07 02 2021.

*What are the time complexities for size?*, W., & IJcken, M. (2020). What are the time complexities for size?. Accessed 7 February 2021, from <https://stackoverflow.com/questions/61390548/what-are-the-time-complexities-for-size>

#### Insertion Sort for Singly Linked List - GeeksforGeeks. (2015). Accessed 7 February 2021, from <https://www.geeksforgeeks.org/insertion-sort-for-singly-linked-list/>