Student Number

Sample Assignment

**Contents**

[Table of Figures 2](#_Toc103704990)

[Introduction 3](#_Toc103704991)

[Main part of assessment 4](#_Toc103704992)

[arrays (1D and 2D), 4](#_Toc103704993)

[records, 4](#_Toc103704994)

[records of arrays, 4](#_Toc103704995)

[queues. 4](#_Toc103704996)

[stacks, 4](#_Toc103704997)

[linked list. 4](#_Toc103704998)

[search 4](#_Toc103704999)

[linear 4](#_Toc103705000)

[binary search 4](#_Toc103705001)

[quicksort 4](#_Toc103705002)

[bubble sort 4](#_Toc103705003)

[deletion from data structures 5](#_Toc103705004)

[insertion into data structures 5](#_Toc103705005)

[Ease of insertion/deletion 5](#_Toc103705006)

[Evaluation against specification. 6](#_Toc103705007)

[Time and space complexity of algorithms used 6](#_Toc103705008)

[Bibliography 7](#_Toc103705009)

[Appendix/Appendices 8](#_Toc103705010)

[Glossary 9](#_Toc103705011)

# Table of Figures

[Figure 1 Required Assignment Content 5](#_tyjcwt)

# Introduction

A business has requested for a card game to be created, inside this document is a discussion of possible methods and decisions behind choosing these functions, examples have been provided through out the document to give the reader an understanding of the process.

# Main part of assessment

Demonstrate an understanding of a range of data structure and

The over all idea of this application is to be able to select how many players are currently playing, which players turn it is would they like to switch their card and who with, the winner is decided by whichever player has the highest card is deemed the winner this is a basic concept that may need altering over the creation before public access as an possible issue is that the second player may always have an edge on the first being able to switch the card back to them

methodologies: discuss data structures – as a min:

## arrays (1D and 2D),

an array is a collection of data

that is all of one data type

it is stored in contiguous memory basic term is side by side

the memory for the array is fixed / static it cannot grow or shrink it isn’t flexible

arrays are very easy to manage using the process called index

examples of arrays are as follows

int [] list

int array values

int [] list <- {1,2, -3,4}

these are declared with the [] or the word ‘array’

the array can be named anything

arrays are 0 index based meaning the first position is 0

so, when it comes to adding cards to the system 0 would be the 2 card for example

and the last element position would be length of array -1 this which for each suit would be the ace card is important to prevent bugs

an example of this would be int s[14] = {2,3,4,5,6,7,8,9,10,10,10,10,11};

but when adding each suit to the arrary it will be more effective to use a 2-d array this would be done by inputing

int s [4] [13] = {{2,3,4,5,6,7,8,9,10,10,10,10,11},

{2,3,4,5,6,7,8,9,10,10,10,10,11}

{2,3,4,5,6,7,8,9,10,10,10,10,11},

{2,3,4,5,6,7,8,9,10,10,10,10,11}};

Which would display the following

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0 | Hearts | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 10 | 10 | 10 | 11 |
| 1 | Diamonds | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 10 | 10 | 10 | 11 |
| 2 | Spades | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 10 | 10 | 10 | 11 |
| 3 | Clubs | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 10 | 10 | 10 | 11 |

Obtaining the data for example the 5 of spades would be as follows

Printf(“%d”,s[2][4]

Another array would be the 3d array, but this method is unnecessary for this project for multiple reasons the first reason is that the values of the cards are the same for each suit which can be also displayed on a 2d array. 3d arrays are mainly used for radar applications providing each access as this project is for a card game a 2d array would be the most viable although 1d arrays would perform more efficiently

Provide evidence on which one is better and why it was decided to use this type

## records,

## records of arrays,

records are a more advanced style of arrays that are able to link more than one type of information together this could be the player and which card they currently have but for this to function correctly with more than one player a second array would be needed to determine how many players are in the game as well as the current card they have.

## queues.

Queues in data structure assist in the method data is transferred, similar to day-to-day life a queue works the first information in the list is sent first whilst the last information is sent last an example of this would be when player one would like to swap their card and also player two would like to swap there card this maintains order this can be used to draw the cards from the top the pile.

## stacks,

stacking data is similar to queues although the first data sent will be the last information processed this known as (LIFO) this would be used when adding cards to a pile the first card added the pile would be at the bottom. This will useful when shuffling the cards

## linked list.

Linked lists are a simple data structure setting a point A to point Z using elements which as able to use any datatype and be processed in either a sorted or unsorted manor this method may seem beneficial but when trying to reach a specific element the process must move from point A to that element this can be unnecessary process and will be slower than most methods, but the positives of using linked is that the data to insert and remove data at point A will always be at constant time no matter how complex or small the input size is, but if you wanted to alter the last element you would have to go from point A to Z just amend the element.

**The relative advantages and disadvantages of these data structures must be discussed for higher marks.**

Discuss algorithms – as a min.

## search

## linear

## binary search

comparison of time complexities

sort

## quicksort

## bubble sort

comparison of time and space complexities

there 2 main categories of data structures with 2 sub categories

## primitive data structures

which are

## integer

## float

## characters

## pointers

# non-primitive data structures

which are

## linear data structures

### arrays

### stack

### queue

### linked-listed

## non-linear data structures

which are

### trees

### graphs

## deletion from data structures

## insertion into data structures

## Ease of insertion/deletion

**into/from different data structures must be discussed for higher marks.**

2. Be able to create and test a range of algorithmic techniques:

Specification developed for the card program.

For a higher grade, the

specification must be precise, allowing each part to be accurately

assessed at the end of the development.

Create the program, showing stages of development and then testing.

For a higher grade, the development stages should show the major

stages of development and an iterative approach

**(exact explanations of how particular fragments of code were added is not required).**

3. Be able to provide detailed technical documentation:

Object Diagram

OO Diagram showing all classes involved in the creation of the

program (attributes and methods must be shown).

Any inheritance or other relations between classes must be shown.

(higher grades must have a discussion of the classes used and why these classes were

used).

Object Listing, with attributes and methods

(higher grades must have a discussion of the objects used and why these objects were used).

Code Listing

(commented, variables must have meaningful names).

Code should show modularity.

4. Be able to critically evaluate the performance of the solutions:

# Evaluation against specification.

What parts of the specification were met and what parts weren’t met

(partially met or not met at all).

For a higher grade, any parts of the specification not met must have a

reason why they were not met

(note that it is not sufficient to just say, this was not met due to time constraints).

# Time and space complexity of algorithms used

with a discussion of

how the time complexity is expected to increase with increasing data

sets.

Discussion of how the program could be extended, for a higher grade,

some discussion of the measures needed to extend the program must

be given.

# Bibliography

# Appendix/Appendices

*This should contain any additional information needed to support your work, but which is not your own work eg an article, copy of a webpage (particularly important if the page is likely to be updated regularly), etc.*