<http://www.yakoub.net/>

<http://srclarke.net>

<https://github.com/jjzirkle-dmacc>

Link to my GitHub: https://github.com/ACROO88/CIS152FinalProject

A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structures provide a means to manage large amounts of data efficiently. efficient data structures are a key to designing efficient algorithms.

Encapsulation is to protect the data fields to access from outside classes. Private data fields are encapsulated so that if we make any object to access to data fields, we should go through the fields methods like its geter and seter. Encapsulation help to protect data from other users.

Inheritance means get some facility from parents to child. Inheritance is to use methods, constructor or any other things from super class to sub classes. By inheriting data fields, methods, constructor we can minimize things to be reoccurred in difference classes and doing things well organized. In java we can’t inherit multiple classes, but we can inherit multiple sub classes. retrieve

|  |  |
| --- | --- |
| Sort Algorithm | Big-O |
| Insertion Sort | 0(n^2) |
| Merge Sort | 0(n log n) |
| Bubble Sort | 0(n^2) |
| Selection Sort | 0(n^2) |
| Quick Sort | 0(n log n) |

Abstraction is something where we can hide the data format like all the abstract methods are the declaration of its defined subclass methods. Simply, by declaring abstract methods we can make an idea of what methods will be defined in subclass. Its like a templet, by declaring abstract methods we can hide relevant data of its object. Because we know we can’t create an instance for abstract class. We can also define methods in abstract class, mostly abstract class is used to collect the common feature of some object so that it will be efficient to use.

Polymorphism means many behaviors. We do the polymorphism through the methods overloading and overwriting even for the constructor. Likewise, we define a method several times in class, but we create different behavior by providing different data type or different parameter. Polymorphism is defined using a method or constructor more than one time in same class and defining same method in different class, so that we will be able to use for different objects.

Stack is system of organizing things where we can push things in and out. But there is a system to push and pop the elements so that we can remove the last push element first out. I think there are lots of set of things in our life where we like to see the matter first which we access last. For example, in the browser when we open some tab in this case if we want se the last access and so on, we will able to see that by pressing tab key. I think, there could be some of use of stack in the development of software. Where we will need to facilitate the last in first out or last in first show.

CANCEL Queue is line or a specific way to organize anything where we follow a way to serve as first come first out or served. Queue is one of the best mechanization to put some activity on next. We follow queue in every place like whether we buy groceries in store or served on phone all kind of services follows the queue basis service even if I pick drugs from Walgreens.

pseudocode for a corresponding program

array of index negotiable

array size

front and tail

enqueue()

check the index available to put elements

increment or add to the tail

increment the size

dequeue()

remove element at front

increment the front to the next element

A **data structure** is a specialized format for organizing and storing **data**. General **data structure** types include the array, the file, the record, the table, the tree, and so on. ... In **computer** programming, a **data structure** may be selected or designed to store **data** for the purpose of working on it with various algorithms.

A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structure provide a means to manage large amounts of data efficiently. Efficient data structure are a key of designing efficient algorithms to maximize the use of memory and completing the brilliant tasks by using artificial intelligence of computer to store, process data.

Principle:

|  |
| --- |
|  |
| First Name : | md |
| Last Name: | ahmed |
| Email: | mahmed4@dmacc.edu |
| Username: | mahmed04 |

General dynamic:

Pas: Riyad100

Ui: mahmed4@dmacc.edu

3 points

**Question 2**

1. What are Michelle's principles of good programming?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| For the toolbar, press ALT+F10 (PC) or ALT+FN+F10 (Mac).   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  | | --- | --- | | [Font family](javascript:;) |  | |  | |  |  | | --- | --- | | [Font size](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | | --- | --- | | [Paragraph](javascript:;) |  | |  | |  |  | | --- | --- | | [Arial](javascript:;) |  | |  | |  |  | | --- | --- | | [3 (12pt)](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  | |  |  | | --- | --- | | [https://dmacc.blackboard.com/webapps/vtbe-tinymce/tiny_mce/plugins/bb_mashup/mashuplogo.png](javascript:;) |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | | 3 | | Path: [p](javascript:;)  Words:15 | |

18 points

**Question 3**

1. The four operations of a computer are the following Blank 1 , Blank 2 , Blank 3 and Blank 4 .

4 points

**Question 4**

1. Match the definition with the data structure.

|  |  |
| --- | --- |
| |  | | --- | |  | |
| Queue |
|  |

|  |
| --- |
| Stack |
|  |

|  |
| --- |
| Linked List |
|  |

|  |
| --- |
| Tree |
|  |

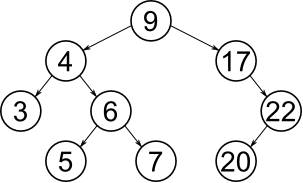
|  |
| --- |
| Graph |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Hash Table | | |  |  | | --- | --- | | A. | A collection of items which are access with first in first out. Basic operations are add and delete. Delete returns the item removed. | | B. | A collection of items mapped by a key, value pair. | | C. | A collection of items which are access with last in first out. Basic operations are add and delete. Delete returns the item removed. | | D. | A linear collection of items, called nodes, each pointing to the next node in sequence. | | E. | A collection of items, called nodes, connected by edges. | | F. | A collection of items, called nodes, with one node as the root and others as child and parent node connected in a hierarchical structure. | |

6 points

**Question 5**

1. Given the following tree, list the in-order of traversal.

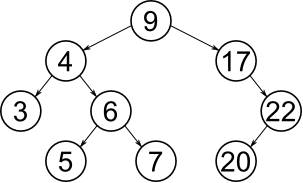


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  | | --- | --- | | [Font family](javascript:;) |  | |  | |  |  | | --- | --- | | [Font size](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | | --- | --- | | [Paragraph](javascript:;) |  | |  | |  |  | | --- | --- | | [Arial](javascript:;) |  | |  | |  |  | | --- | --- | | [3 (12pt)](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  | |  |  | | --- | --- | | [https://dmacc.blackboard.com/webapps/vtbe-tinymce/tiny_mce/plugins/bb_mashup/mashuplogo.png](javascript:;) |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | |  | | Path: [p](javascript:;)  Words:0 | |

9 points

**Question 6**

1. Given the following tree, list the pre-order of traversal.

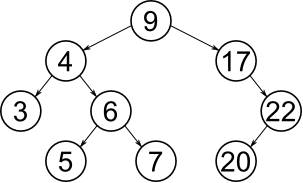


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  | | --- | --- | | [Font family](javascript:;) |  | |  | |  |  | | --- | --- | | [Font size](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | | --- | --- | | [Paragraph](javascript:;) |  | |  | |  |  | | --- | --- | | [Arial](javascript:;) |  | |  | |  |  | | --- | --- | | [3 (12pt)](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  | |  |  | | --- | --- | | [https://dmacc.blackboard.com/webapps/vtbe-tinymce/tiny_mce/plugins/bb_mashup/mashuplogo.png](javascript:;) |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | |  | | Path: [p](javascript:;)  Words:0 | |

9 points

**Question 7**

1. Given the following tree, list the post-order of traversal.



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  | | --- | --- | | [Font family](javascript:;) |  | |  | |  |  | | --- | --- | | [Font size](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | | --- | --- | | [Paragraph](javascript:;) |  | |  | |  |  | | --- | --- | | [Arial](javascript:;) |  | |  | |  |  | | --- | --- | | [3 (12pt)](javascript:;) |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  | | --- | --- | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  | |  |  | | --- | --- | | [https://dmacc.blackboard.com/webapps/vtbe-tinymce/tiny_mce/plugins/bb_mashup/mashuplogo.png](javascript:;) |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | |  | | Path: [p](javascript:;)  Words:0 | |

9 points

**Question 8**

1. Match the sorting definitions

|  |  |
| --- | --- |
| |  | | --- | |  | |
| Bubble Sort |
|  |

|  |
| --- |
| Insertion Sort |
|  |

|  |
| --- |
| Merge Sort |
|  |

|  |
| --- |
| Selection Sort |
|  |

|  |
| --- |
| Quicksort |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Heapsort | | |  |  | | --- | --- | | A. | A simple sorting algorithm, it starts at the beginning and compares the first two items. If the first is greater than the second, it swaps them. It continues comparing and appropriately swapping for each pair of adjacent elements to the end of the items. It begins again with the first two items, repeating until no swaps occur on the last pass. | | B. | This sort joins already sorted lists into a new sorted list. It compares the first item of each list and selects the smaller (or larger item) to join in the new list. This continues until one list is empty when the remaining list is simply added. | | C. | This first finds the minimum value item and swaps it with the item in the first position. This is repeated on the remainder for n iterations, where n is the size of the list. | | D. | This is a divide and conquer algorithm which partitions a list an item called a *pivot.* All items smaller than the pivot are moved before it and all greater elements are moved after it | | E. | This is a simple sorting algorithm that takes items from the list one by one and places them in their correct position into a new sorted list. | | F. | In this method, it determines the largest (or smallest) item of the list, moves it to the beginning (or end) of the list. This continues with the rest of the list, but accomplishes this task efficiently by using a data structure which is a special type of binary tree. | |

6 points

**Question 9**

1. Which of these sorting algorithms is most efficient?

|  |  |  |
| --- | --- | --- |
|  |  | Merge Sort |
|  |  | Bubble Sort |
|  |  | Selection Sort |
|  |  | Insertion Sort |

1 points

Click Save and Submit to save and submit. Click Save All Answers to save all answers.

**Modularity**-- This principle addresses writing pieces of code in Top-Down Design so they are re-usable. In an Object-Oriented language this means writing classes and utilizing encapsulation.

**Efficiency**-- This principle means efficient in memory and time usage. Algorithms should be efficient, you should not use more memory than necessary while allowing for scalability.

**Robustness** -- This principle means a program should not break, but since we cannot write bug-free code, the code handles errors when possible. When it cannot, the program should exit gracefully (ie, no blue screen of death)

**Usability** -- This principle addresses that a program should do what it was intended to do. Programmers may concentrate on bells and whistles and run into scope creep where they never accomplish the assigned task. Also, a program should be user-friendly and intuitive for the end-user (programmers often make code that is intuitive to themselves.)

**Should be** **Readable** -- This principles indicates the need for self-commenting code, appropriate use of indentation and whitespace, appropriate naming of variables and the inclusion of comments through out.

**Elegant** -- This principle addresses the idea of beautiful code. Programmers should be consistent in their style that leads to readable code. Additionally, shorter code and efficient algorithms lend to elegant code.

Define HashMap, Hash table and compare the two in 100 words or more. This is worth 15 points.

HashMap:

HashMap is a set of map data structure which stores data in the form of key-value pair. We can implement hash map in an associative array, where we will able to build a structure that can map keys to the values.

HashTable:

A hash table uses a hash function to compute an index into an array of buckets or slots, from which the correct value can be found. synchronized Objects typically perform better than unsynchronized ones in hash table. It does not take any null value.

Difference between hash table and hash table:

|  |  |
| --- | --- |
| **Hash Map** | **Hash table** |
| Introduced in 1.2 version | This since java was introduced |
| It is not thread safe and unsynchronized | It is thread safe and synchronized |
| It is fast | It is slow |
| Works with single thread | Work with multiple threads |
| Allows one null key or value | Does not allow null key or value |
|  |  |