****

**NAME: MUTAI K. BRUCE**

**REG NO: SB06/SR/MN/11103/2020**

**TITTLE: DESIGN AND ANALYSIS**

**CODE: COM3106**

**TASK: CAT ONE**

1. **Discuss the basic operations and representations of the following derived data types (20 marks).**
2. **Arrays**

An array is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier.

There are two types of arrays;

1. **Two dimensional array.**

Syntax; DataType ArrayName [row\_size] [column\_size].

Example: int arr[4] [4].

1. **Three dimensional array.**

When we require to create two or more tables of the elements to declare the array elements, then in such a situation we use three-dimensional arrays.

Syntax: Datatype ArrayName [size1] [size2] [size3].

Example: int a[5] [5] [5].

**Operations on Array data structures.**

**Insertion:** Adds an element at the given index.

**Deletion:** Deletes an element at the given index.

**Search:** Searches an element using the given index or by the value.

1. **Lists**

A list is an ordered data structure with elements separated by a comma and enclosed within square brackets.

Example;

|  |
| --- |
| List 1= [2,3,4,5,6] |

|  |
| --- |
| List 2= [‘Python’, ‘is’, ‘Awesome’] |

**Operations on lists data structure.**

**Insertion:** Adds an element at the beginning of the list.

**Deletion:** Deletes an element at the beginning of the list.

**Display:** Displays the complete list.

**Search:** searches an element using the given key.

1. **Stacks.**

Stack is a linear data structure which follows a particular order in which the operations are performed.

The order may be **LIFO** (Last in First Out) or **FILO** (First in Last Out).

|  |
| --- |
| **6** |
| 5 |
| 4 |
| 3 |
| 2 |
| 1 |

**Operations on stack data structure.**

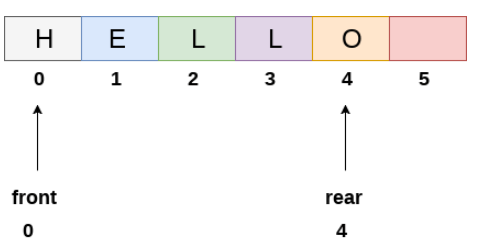
* **push():** to insert an element into the stack
* **pop():**to remove an element from the stack
* **top():** Returns the top element of the stack.
* **isEmpty():**returns true is stack is empty else false
* **size():**returns the size of stack

1. **Queues.**

A queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out **(FIFO)** order.

Queue is defined as a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end.

Representation of a queue;



**Operations on queue.**

**Enqueue operation:**

Enqueue means inserting an element in the queue. Anew element in a queue is inserted at the back of the queue.

**Dequeue operation:**

Dequeue means removing an element from the queue. Since queue follows the FIFO principle we need to remove the element of the queue which was inserted at first. Naturally, the element inserted first will be at the front of the queue so we will remove the front element and let the element behind it be the new front element.

**Front operation:**

This is similar to the peek operation in stack, it returns the value of the element at the front without removing it.

**IsEmpty:**

Checks if the queue is empty. To prevent performing operations on an empty queue, the programmer is required to internally maintain the size of the queue which will be updated during enqueuer and dequeue operations accordingly.

IsEmpty() conventionally returns a Boolean value: True if size is 0, else False.

1. **Discuss hashing as used in data structures citing its operations.**

Hashing in data structures uses hash tables to store the key-value pairs. The hash table then uses the hash function to generate an index. Hashing uses this unique index to perform insert, update, and search operations. It can be defined as a bucket where the data are stored in an array format.

Hashing in the data structure is a technique of mapping a large chuck of data into small tables using a hashing function. It is also known as the message digest function. It is a technique that uniquely identifies a specific item from a collection of similar items.

It uses hash tables to store the data in an array format. Each value in the array has been assigned a unique index number. Hash tables use a technique to generate these unique index numbers for each value stored in an array format. This technique is called the hashtechnique.

You only need to find the index of the desired item, rather than finding the data.

**Operations of hash table.**

**Search():** Searches an element in a hash table.

**Insert():** Inserts an element in a hash table.

**Delete():** Deletes an element from a hash table.

**References;**

1. Data Structures and Algorithms Made Easy:

Data Structures and Algorithmic Puzzles by

Narasimha Karumanchi

1. Adavanced Data Structure by Peter Brass
2. Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein