Lecture 3 Array and Linked List

Class	Cohort 2 Year 1
Created	@Oct 7, 2020 5:49 PM
Materials	
Reviewed	☐
Type	

Introduction To Arrays

Challenges:

1. Remove Even Integers From An Array
   - Implement a function removeEven(int *& Arr, int size) which takes an array arr and its size and removes all the even elements from a given array.
2. Merge Two Sorted Arrays
   - Implement a function mergeArrays(int arr1[], int arr2[], int arr1Size,int arr2Size) which merges two sorted arrays into another sorted array. Definition of the function is given.
3. Find Two Numbers That Add up to "Value"
   - Implement a function findSum(int arr[], int value, int size) which takes an array arr, a number value and size of the array as input and returns an array of two numbers that add up to value. In case there is more than one pair in the array containing numbers that add up to value, you are required to return only one such pair. If no such pair found then simply return the array.
4. Array of Products of all Elements
   - Implement a function, findProduct(int arr[], int size) which takes an array arr and its size as an input and returns an array so that each

index has a product of all the numbers present in the array except the number stored at that index.

5. Find Minimum Value in an Array
    - Implement a function findMinimum(int arr[], int size) which takes an array arr and its size and returns the smallest number in the given array.
6. Find First Unique Integer in an Array
    - Implement a function, findFirstUnique(int arr[], int size) which takes an array and its size as input and returns the first unique integer in the array. The function returns -1 if no unique number is found.
7. Find Second Maximum Value in an Array
    - Implement a function findSecondMaximum(int arr[], int size) which takes an array arr and its size as input and returns the second maximum element in the array. If no such element found then return secondmax variable.
8. Right Rotate an Array by 1
    - Implement a function rightRotate(int arr[], int size) which takes an array arr and rotate it right by 1. This means that the right-most elements will appear at the left-most position in the array.
9. Rearrange Positive & Negative Values
    - Implement a function reArrange(int arr[], int size) which takes an array arr and its size as input and rearranges the elements such that all the negative elements appear on the left and positive elements appear at the right.
10. Rearrange Sorted Array in Max//Min Form
    - Implement a function maxMin(int arr[], int size) which takes a sorted array arr and its size and will re-arrange the elements of a sorted array such that the first position will have the largest number, the second will have the smallest, and the third will have second largest and so on. In other words, all the even-numbered indices will have the largest numbers in the array in descending order and the odd-numbered indices will have the smallest numbers in ascending order.
11. Maximum Sum Subarray

- Given an integer array and its size, return the maximum subarray sum. The array may contain both positive and negative integers and is unsorted.

Introduction To Linked Lists

Singly Linked Lists(SLL)

```
class Node {
public:
  int data; //Data to store (could be int,string,object etc)
  Node* nextElement;  //Pointer to next element

  Node(){
  //Constructor to initialize nextElement of newly created Node
    nextElement=nullptr;
  }
};

class LinkedList {
public:
  Node* head;  // pointing to start of the list

  LinkedList(){
    head = nullptr;
  }
};
```

Basic Operations on Linked Lists

The primary operations which are generally a part of the `LinkedList` class are listed below:

- `insertAtTail(data)` - inserts an element at the end of the linked list
- `insertAtHead(data)` - inserts an element at the start/head of the linked list
- `delete(data)` - deletes an element with your specified value from the linked list
- `deleteAtHead()` - deletes the first element of the list
- `search(data)` - searches for an element in the linked list
- `getHead()` - returns head of the linked list
- `isEmpty()` - returns true if the linked list is empty

Insertion

1. Insert at Head
2. Insert at Tail
3. Insert at the N-th index

Deletion in a Singly Linked List

1. Deletion at the head
2. Deletion by value
3. Deletion at the tail

To cover 1. rest in exercise

Doubly Linked Lists (DLL)

```
class Node {
public:
  int data; //Data to store (could be int,string,object etc)
  Node * nextElement;  //Pointer to next element
  Node * previousElement; //pointer to previous element
  Node(){
  //Constructor to initialize nextElement of newlyCreated Node
    nextElement=nullptr;
    previousElement=nullptr;
  }
};
```