LAB Exam I

CS211 – Data Structures & Algorithms  
Usman Institute of Technology  
Fall 2020

1. Write a recursive function **CoconutSearch** that takes three parameters: an array, starting position, and the number to search. It returns the location of a number if exists in the array, otherwise returns -1.

The function works as per the following recursive definition:

* if e is at the location loc in the array then returns 1oc,
* Otherwise, search again at loc+1
* If loc is greater than number of elements than returns -1

**def** **CoconutSearch**(array, loc, e):

// your code goes here

Example:

array = [1,2,3,4,5]

print(CoconutSearch(array,0, 2))

print(CoconutSearch(array,0, 6))

#the function should return

1

-1

1. Write a function **RotateLeft** that rotates an array of size n to the left. The function takes two parameters an integer d which is the amount to rotate by and an array. The function should return the rotated array.

**def** **RotateLeft**(array, d):

// your code goes here

Example:

array = [1,2,3,4,5]

d = 2 #two rotations

print(RotateLeft(array,d))

#the function should return

[3,4,5,1,2]

1. Create a class DStack to implement two stacks in one array A[1….n] in such a way that neither stack overflows unless the total number of elements in both stacks together is size n. The Push and Pop operations should run in O(1).

The class should implement following functions:

* 1. Write a **constructor** that takes an integer ‘size’ as argument and creates an array for ‘size’ number of elements.
  2. Write a function **Push** that inserts an element for the stack ‘s’.
  3. Write a function **Pop** that removes an element for the stack ‘s’.
  4. Write a function **Peek** that returns an element for the stack ‘s’.
  5. Write a function **IsEmpty** that returns true stack ‘s’ is empty.
  6. Write a function **Count** that returns the number of elements in stack ‘s’

class **DStack**:

**# size = number of elements**

**def \_\_init\_\_(self, size):**

// your code goes here

**# s = 1 or 2**

**def** **Push**(self, s, value):

// your code goes here

**# stack = 1 or 2**

**def** **Pop**(self, s):

// your code goes here

**# s = 1 or 2**

**def** **Peek**(self, s):

// your code goes here

**# s = 1 or 2**

**def** **IsEmpty**(self, s):

// your code goes here

**# s = 1 or 2**

**def** **Count**(self, s):

// your code goes here

Example:

ds = DStack(10)

ds.Push(1,10) #push 10 in stack 1

ds.Push(2,20) #push 20 in stack 2

ds.Push(1,30) #push 30 in stack 1

print(ds.Pop(1)) #pop from stack 1

print(ds.Peek(2)) # peek from stack 2

#the function should return

30

20

1. Create a class **Deque** and implement the functions of double ended queue. All functions should run in O(1).
   1. Write a function **InsertFront** that inserts an element at the front of the Deque.
   2. Write a function **GetFront** that returns the front item from the Deque.
   3. Write a function **DeleteFront** that removes an item from the front of Deque.

class **Deque**:

def \_\_init\_\_(self):

// your code goes here

**def** **InsertFront**(self, value):

// your code goes here

**def** **GetFront**(self):

// your code goes here

**def** **DeleteFront**(self):

// your code goes here

1. Create a class **DList** and add the following functions. All functions should run in O(n).
2. Write a function **Insert** that takes an argument x. Where x is an integer value. The function should insert the value x in a sorted doubly linked list. You have to make sure that the list should remain sorted after inserting the value of x.
3. Write a function **IsSorted** that checks if the doubly Linked List is sorted or not. The function should return **True** if the List is sorted otherwise it should return **False**.
4. Write a function **DeleteDuplicates** that deletes duplicate values from the linked list. The function should return a sorted list after deleting the duplicate values.

class **DNode**:

def \_\_init\_\_(self,value):

self.value = value

self.next = None

self.previous = None

class **DList**:

def \_\_init\_\_(self):

self.head = None

self.tail = None

**def** **Insert**(self,x,j):

// your code goes here

**def** **IsSorted**(self):

// your code goes here

**def** **DeleteDuplicates**(self):

// your code goes here