Data Structures and Algorithms

Roll No: AA.SC.P2MCA2107434

DSA LAB ASSIGNMENT -1

Task

AIM 1: Understanding the concept of Array and its Applications (5 points)

An array is a collection of similar data elements. These data elements have the same data type. The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript). The subscript is an ordinal number which is used to identify an element of the array.

Operations on Arrays

There are a number of operations that can be preformed on arrays. These operations include:

• Traversing an array Inserting an element in an array

• Searching an element in an array

• Deleting an element from an array

• Merging two arrays Sorting an array in ascending or descending order

1) Implement a program for inserting a new element to the specified position of an array.

2) Implement a program for deleting an element from the specified position of an array.

3) Implement a program for sorting a given set of numbers.

AIM 2: Understanding the concepts of stack, its implementations and applications.(5 points)

• Stack is linear data structure in which addition or deletion takes place at the same end. This end is called the top of stack. Examples of stack are: Stack of plates, Stack of Books etc. Stack is a sequence of items, which can be added and removed from one end only.

• Stack is known as LIFO (last in first out).

• Insert Operation (PUSH) Stacks can be implemented using arrays by defining a structure containing an array and variable to indicate the position of top of stack. PUSH – add data x to stack Increment top and then set data[top]= x

• Delete Operation (POP) POP-remove and return data from stack Return data[top] and decrement top

1) Implement a program for creating a new stack, adding element to the stack, removing elements from stack.

2) Implement a program to reverse a given string using stack.

AIM-1

In [1]:

```
import numpy as np
np.set_printoptions(suppress=True)
```

In [2]:

```
x = [5,8,6,9,2]
arr = np.array(x,float)
```

In [3]:

```
def insert():
    global arr
    print("Input position")
    k = int(input())
    if(k>=0):
        print("Input number to insert")
        num = float(input())
        x.insert(k, num)
        arr = np.asarray(x)
    else:
        print("Index out of range")
```

In [4]:

```
flag_input = True
while(flag_input):
    insert()
    print("Want to continue inserting elements? Y/N")
    cont = str(input())
    if(cont == 'N'):
        flag_input = False
```

```
Input position
2
Input number to insert
4
Want to continue inserting elements? Y/N
N
```

In [5]:

```
arr # 4 added
```

Out[5]:

```
array([5., 8., 4., 6., 9., 2.])
```

In [6]:

```
def delete():
    global arr
    print("Input position")
    k = int(input())
    if(k>=0):
        arr1 = np.delete(arr, k)
        arr = arr1
    else:
        print("Index out of range")
```

In [7]:

```
flag_delete = True
while(flag_delete):
    delete()
    print("Want to continue deleting elements? Y/N")
    cont = str(input())
    if(cont == 'N'):
        flag_delete = False
```

```
Input position
2
Want to continue deleting elements? Y/N
N
```

In [14]:

```
arr # 4 gone
```

Out[14]:

```
array([2., 5., 6., 8., 9.])
```

In [9]:

```
def bubble_sort(nums):
    swapped = True
    while swapped:
        swapped = False
        for i in range(len(nums) - 1):
            if nums[i] > nums[i + 1]:
                nums[i], nums[i + 1] = nums[i + 1], nums[i]
                swapped = True
    return nums
```

In [10]:

```
arr = np.array(bubble_sort(list(arr)))
arr
```

Out[10]:

```
array([2., 5., 6., 8., 9.])
```

AIM-2

In [11]:

```
def createStack():
    stack = []
    return stack
```

```
def isEmpty(stack):
    return len(stack) == 0

def push(stack, item):
    stack.append(item)
    print(item, " pushed to stack ")

def pop(stack):
    if (isEmpty(stack)):
        return None
    return stack.pop()

def print_stack(stack):
    print("Stack = ", stack)
```

In [12]:

```
stack = createStack()
push(stack, 42)
push(stack, 55)
push(stack, 22)
print(pop(stack), " popped from stack")
push(stack, 27)
push(stack, 56)
print(pop(stack), " popped from stack")
push(stack, 7)
print_stack(stack)
```

```
42  pushed to stack
55  pushed to stack
22  pushed to stack
22  popped from stack
27  pushed to stack
56  pushed to stack
56  popped from stack
7  pushed to stack
Stack =  [42, 55, 27, 7]
```

In [13]:

```
stack = createStack()
push(stack, 'C')
push(stack, 'H')
push(stack, 'A')
push(stack, 'I')
push(stack, 'R')
print(pop(stack), pop(stack), pop(stack), pop(stack), pop(stack))
```

```
C  pushed to stack
H  pushed to stack
A  pushed to stack
I  pushed to stack
R  pushed to stack
R I A H C
```

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js