**INDIAN SCHOOL AL GHUBRA**

**CLASS XII COMPUTER SCIENCE DATE: 12/7/2020**

**LAB ASSIGNMENT 18**

Write a program to implement a stack for the given book details (Bookno, Bookname and Cost). ie each item node of the stack contains three types of information. Implement Push, Pop and Display Operations. Top should be a global variable which will display the size of the stack at any point of time. The program should be Menu driven which should be terminated based on the user’s choice.

**LAB ASSIGNMENT 19**

Write a program in Python to reverse the contents of a file using stack.

Examples:

Input :

This is the program

to reverse the

content of a file

using stack

Output :

using stack

content of a file

to reverse the

This is the program

Input :

1

2

3

4

5

Output :

5

4

3

2

Hint:

* Create an empty stack.
* One by one push every line of the file to the stack.
* One by one pop each line from the stack and put them back to the file.

**LAB ASSIGNMENT 19**

Write a program to delete all even elements from a stack

Given a stack with n elements, the task is to remove all the elements of the stack without affecting the order of elements.

Examples:

Input : s = 16 <- 15 <- 29 <- 24 <- 19 (TOP)

Output: 19 29 15

19 29 15 is the order of odd elements in which

they will be popped from the given stack.

Input : s = 1 <- 2 <- 3 <- 4 <- 5 (TOP)

Output: 5 3 1

Approach:

1. Create a temporary stack temp and start popping the elements of the given stack s.

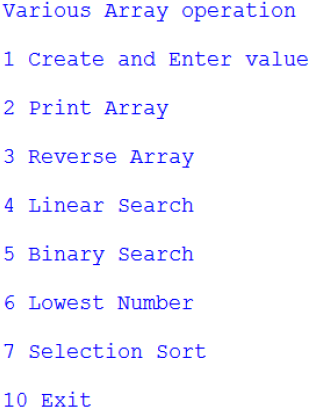
2. For every popped element say val, if val % 2 == 1 then push it to temp.

3. At the end of step 2, temp will contain all the odd elements from s but in reverse order.

4. Now, to get the original order, pop every element from temp and push it to s.

**LAB ASSIGNMENT 20**

Write a Program to enter the numbers and perform Linear Search, Binary Search, Lowest Number and Selection Sort using list/array code. A sample output screenshot is given bellow.



**LAB ASSIGNMENT 21**

Write a program to perform insert and delete operations on a Queue containing Member details as given on the following definition of itemnode

MemberNo integer

MemberName String

Age integer

**LAB ASSIGNMENT 22**

Write a program that implements three queues namely VIP, Balcony and Regular. The program accepts the tokenID with it priority from the user eg:

Enter tokenID: ABC123

Priority(Highest/Normal/lowest(H/N/L): H

As per the priority entered, the element is added in the corresponding queue.

A menu offers the following options

1. Insert tokenId
2. Search for an Id
3. Change Priority

|  |
| --- |
| # Creating Stack class (LIFO rule)  class Stack:        def \_\_init\_\_(self):            # Creating an empty stack          self.\_arr = []        # Creating push() method.      def push(self, val):          self.\_arr.append(val)        def is\_empty(self):            # Returns True if empty          return len(self.\_arr) == 0        # Creating Pop method.      def pop(self):            if self.is\_empty():              print("Stack is empty")              return            return self.\_arr.pop()    # Creating a function which will reverse  # the lines of a file and Overwrites the  # given file with its contents line-by-line  # reversed  def reverse\_file(filename):        S = Stack()      original = open(filename)        for line in original:          S.push(line.rstrip("\n"))        original.close()          output = open(filename, 'w')        while not S.is\_empty():          output.write(S.pop()+"\n")        output.close()      # Driver Code  filename = "GFG.txt"    # Calling the reverse\_file function  reverse\_file(filename)    # Now reading the content of the file  with open(filename) as file:          for f in file.readlines():              print(f, end ="") |

**Output:**

Ths is a World of Geeks.

Welcome to GeeksforGeeks.

# Delete all even elements from a stack

Given a stack with **n** elements, the task is to remove all the elements of the stack without affecting the order of elements.

**Examples:**

***Input :****s = 16 <- 15 <- 29 <- 24 <- 19 (TOP)****Output:****19 29 15  
19 29 15 is the order of odd elements in which  
they will be popped from the given stack.*

***Input :****s = 1 <- 2 <- 3 <- 4 <- 5 (TOP)****Output:****5 3 1*

**Approach:**

1. Create a temporary stack **temp** and start popping the elements of the given stack **s**.
2. For every popped element say **val**, if **val % 2 == 1** then push it to **temp**.
3. At the end of step 2, **temp** will contain all the odd elements from **s** but in reverse order.
4. Now, to get the original order, pop every element from **temp** and push it to **s**.

// C++ implementation of the approach

#include <stack>

#include <iostream>

#include <stdio.h>

using namespace std;

// Utility function to print

// the contents of a stack

static void printStack(stack<int> s)

{

while (!s.empty())

{

cout << s.top() << " ";

s.pop();

}

}

// Function to delete all even

// elements from the stack

static void deleteEven(stack<int> s)

{

stack<int> temp;

// While stack is not empty

while (!s.empty())

{

int val = s.top();

s.pop();

// If value is odd then push

// it to the temporary stack

if (val % 2 == 1)

temp.push(val);

}

// Tranfer the contents of the temporary stack

// to the original stack in order to get

// the original order of the elements

while (!temp.empty())

{

s.push(temp.top());

temp.pop();

}

// Print the modified stack content

printStack(s);

}

// Driver Code

int main()

{

stack<int> s;

s.push(16);

s.push(15);

s.push(29);

s.push(24);

s.push(19);

deleteEven(s);

return 0;

}

# Delete middle element of a stack

Given a stack with push(), pop(), empty() operations, delete middle of it without using any additional data structure.

Input : Stack[] = [1, 2, 3, 4, 5]

Output : Stack[] = [1, 2, 4, 5]

Input : Stack[] = [1, 2, 3, 4, 5, 6]

Output : Stack[] = [1, 2, 4, 5, 6]

The idea is to use recursive calls. We first remove all items one by one, then we recur. After recursive calls, we push all items back except the middle item.

# Python3 code to delete middle of a stack

# without using additional data structure.

# Deletes middle of stack of size

# n. Curr is current item number

class Stack:

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

self.items = []

def isEmpty(self):

return self.items == []

def push(self, item):

self.items.append(item)

def pop(self):

return self.items.pop()

def peek(self):

return self.items[len(self.items)-1]

def size(self):

return len(self.items)

def deleteMid(st, n, curr) :

# If stack is empty or all items

# are traversed

if (st.isEmpty() or curr == n) :

return

# Remove current item

x = st.peek()

st.pop()

# Remove other items

deleteMid(st, n, curr+1)

# Put all items back except middle

if (curr != int(n/2)) :

st.push(x)

# Driver function to test above functions

st = Stack()

# push elements into the stack

st.push('1')

st.push('2')

st.push('3')

st.push('4')

st.push('5')

st.push('6')

st.push('7')

deleteMid(st, st.size(), 0)

# Printing stack after deletion

# of middle.

while (st.isEmpty() == False) :

p = st.peek()

st.pop()

print (str(p) + " ", end="")

# This code is contributed by

# Manish Shaw (manishshaw1)

# Check if a given number is a Perfect square using Binary Search

Check if a given number **N** is a perfect square or not. If yes then return the number of which it is a perfect square, Else print -1.

**Examples:**

***Input:****N = 4900****Output****70****Explanation:*** *4900 is a perfect square number of 70 because 70 \* 70 = 4900*

***Input:****N = 81****Output:****9****Explanation:*** *81 is a perfect square number of 9 because 9 \* 9 = 81*

**Approach:** To solve the problem mentioned above we will use the [Binary Search Algorithm.](https://www.geeksforgeeks.org/binary-search/)

* Find the mid element from the start and last value and compare the value of the square of mid(mid\*mid) with N.
* If it is equal then return the mid otherwise check if the square(mid\*mid) is **greater than N** then recursive call with the same start value but changed last to mid-1 value and if the square(mid\*mid) is **less than the N** then recursive call with the same last value but changed start value.
* If the N is not a square root then return -1.

# Python3 program to check if a

# given number is perfect

# square using Binary Search

# Function to check for

# perfect square number

def checkPerfectSquare(N, start, last):

# Find the mid value

# from start and last

mid = int((start + last) / 2)

if (start > last):

return -1

# Check if we got the number which

# is square root of the perfect

# square number N

if (mid \* mid == N):

return mid

# If the square(mid) is greater than N

# it means only lower values then mid

# will be possibly the square root of N

elif (mid \* mid > N):

return checkPerfectSquare(N, start, mid - 1)

# If the square(mid) is less than N

# it means only higher values then mid

# will be possibly the square root of N

else:

return checkPerfectSquare(N, mid + 1, last)

# Driver code

N = 65

print (checkPerfectSquare(N, 1, N))

# This code is contributed by PratikBasu

# Reverse individual words

Given a string str, we need to print reverse of individual words.

**Examples:**

Input : Hello World

Output : olleH dlroW

Input : Geeks for Geeks

Output : skeeG rof skeeG

**Approach:** We use a stack to push all words before space. As soon as we encounter a space, we empty the stack.

# Python3 program to reverse individual words

# in a given string using STL list

# reverses individual words of a string

def reverserWords(string):

st = list()

# Traverse given string and push all characters

# to stack until we see a space.

for i in range(len(string)):

if string[i] != " ":

st.append(string[i])

# When we see a space, we print

# contents of stack.

else:

while len(st) > 0:

print(st[-1], end= "")

st.pop()

print(end = " ")

# Since there may not be space after

# last word.

while len(st) > 0:

print(st[-1], end = "")

st.pop()

# Driver Code

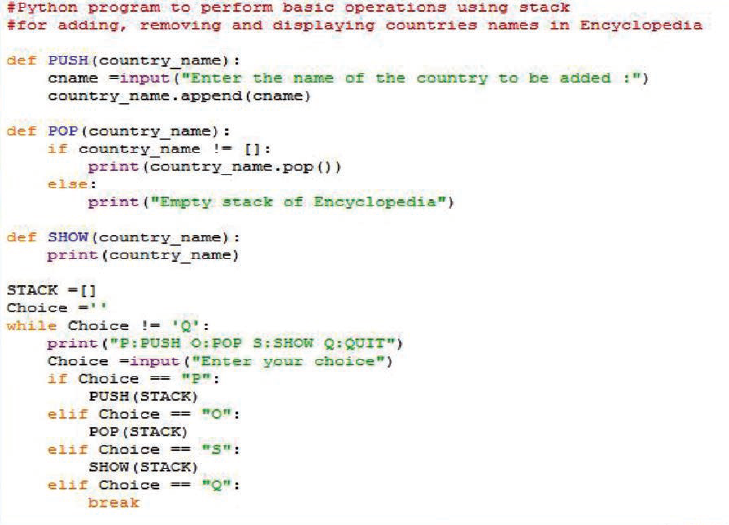
if \_\_name\_\_ == "\_\_main\_\_":

string = "Geeks for Geeks"

reverserWords(string)

Infopedia is an online encyclopedia which stores detailed information about various countries. In order to provide efficient processing for retrieval and display of names of countries, it has to be digitized, which requires handling additions and deletions through data structures in Python, primarily stacks and queues.

Write a program constituting methods in Python to add, display and remove a name from a given stack of names of countries.



FLYAIR Airlines has grown big and touches more than 140 destinations worldwide, offering excellent service to its passengers. To provide latest information to its customers at the click of a button, the company offers computerized processing of passenger details.

Write a Python program comprising methods/functions to add or delete a passenger’s name from the list of passengers, considering them as insert and delete operations of the Queue data structure.

