BRAINWARE UNIVERSITY
398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125
Laboratory Report / Workshop / Assignment Submission
Session - 2024 - 25

Lab Report No: -

Topic Title: -

Name of the Department: - Cyber Science & Technology

Programme Name: - B.Sc (H) ANCS 2024

Semester / Year: - Semester-2, 2024-2025

Course Code: - VAC09009

Course Name: - Python Programming Lab

Name of the Student: - Rupankar Chakraborty.

Roll No: -

Registration No: -

Student Code: -BWU/BNC/24/ 157

09.04.25

Assignment-10 D:09.04.25

Stack: (LIFO)

A stack is a part of data structure which follows the classical linear data structure it basically uses the concept of LIFO as it's principal method to solve a problem.

Operations of Stack:

1. push()
   It inserts the items or values at the top.
2. pop()
   It returnes or removes the values from top.
3. peek()
   Views the value at the top.
4. is_empty()
   checks if the stack is empty or not.

[Note: Class: Python is an object oriented programming language, in python almost everything is concidered as an object with it's properties and methods. A class is like an object constructor or it's basically a blueprint for creating an object. To create class we use the keyword class. The "__init__()" function all classes have a function called init(), which is always executed when the class is being initiated. use the init() function to assign values to object properties or other operations that are necessery to do when the object is being initiated.

```
class stack:
    def __init__(self):
        self.stack = []
    def push(self, item):
        self.stack.append(item)
    def pop(self):
        return self.stack.pop() if self.stack else None
    def peek(self):
        return self.stack[-1] if self.stack else None
    def is_empty(self):
        return len(self.stack) == 0

s = Staek()
s.push(1)
s.push(2)
s.push(3)

print(s.peek())
print(s.pop())
print(s.is_empty())
```

Output: 3
3
False

Write a python programme to manage a student's marks accross 3 semesters for each semester

i. input number of subjects.

ii. For each subject:

ii. A. Enter marks

B. If Marks < 50, allow retake until the student passes (≥ 50), using a stack (collections.deque) to ~~Dec~~ store each attempt.

c. Final mark = latest (top of stack) attempt.

After each semester, display total, average, and grade. After 3 semesters, show overall total, average and final grade.

Grade Scale: A+ (≥ 90), A (≥ 80), B (≥ 70), C (≥ 60)

```
    # Calculate total and average
    total = sum(subject_marks)
    avg = total / len(subject_marks)
    grade = calculate_grade(avg)

    print(f"\n---Semester {sem_num} Summary---")
    print(f"Total : {total}")
    print(f"Average : {avg:.2f}")
    print(f"Grade : {grade}")

    return total, len(subject_marks)
def student_marks_3sem_with_retakes():
    name = input("Enter student name:")
    roll = input("Enter roll number")

    print(f"\n---Entering Marks for 3 Semesters with Retake
Option---")
    grand_total = 0
    grand_count = 0
    for sem in range(1,4): # 3 semesters
        sem_total, sem_count = process_semester(sem)
        grand_total += sem_total
        grand_count += sem_count
    overall_avg = grand_total / grand_count
    overall_grade = calculate_grade(overall_avg)
    print(f"\n====Overall Result ====")
    print(f"Name : {name}")
    print(f"Roll No : {roll}")
    print(f"Total Marks: {grand_total}")
    print(f"Average : {overall_avg:.2f}")
    print(f"Final Grade : {overall_grade}")

#Run it
student_marks_3sem_with_retakes()
```

D(≥50), F(<50)

```
from collections import deque
def calculate_grade(avg):
    if avg >= 90: return 'A+'
    elif avg >= 80: return 'A'
    elif avg >= 70: return 'B'
    elif avg >= 60: return 'C'
    elif avg >= 50: return 'D'
    else: return 'F'
def process_semester(sem_num):
    num_subjects = int(input(f"\nEnter number of
subjects for Semester {sem_num}: "))
    subject_marks = []

    for i in range(num_subjects):
        subject_stack = deque()
        subject_name = input(f"\nEnter name of Subject
{i+1} (Sem {sem_num}): ")
        # First Attempt
        mark = float(input(f"Enter marks for {subject_
name}: "))
        subject_stack.append(mark)

        #If failed, allow re-attempt(s)
        while subject_stack[-1] < 50:
            print(f"⚠ Failed in {subject_name} (Marks: {subject
_stack[-1]}), enter new marks.")
            mark = float(input(f"Re-enter marks for
{subject_name}: "))
            subject_stack.append(mark)

        # Final mark is top of stack
        final_mark = subject_stack[-1]
        print(f"☑ Final mark for {subject_name}: {final_
mark}")
        subject_marks.append(final_mark)
```

Output: Enter student name: Rupankar
Enter roll number:

---Entering Marks for 3 Semesters with Retake Option ---
Enter number of subjects for Semester 1: 3
Enter name of Subject 1 (Sem 1): DBMS
Enter marks for DBMS: 60
☑ Final mark for DBMS: 60.0
Enter name of Subject 2 (Sem 1): Network
Enter marks for Network: 75
☑ Final mark for Network: 75.0
Enter name of Subject 3 (Sem 1): Soft Skills
Enter marks for Soft Skills: 78
☑ Final mark for Soft Skills: 78.0
--- Semester 1 Summary ---
Total: 213.0
Average: 71.00
Grade: B
Enter number of subjects for Semester 2: 2
Enter name of Subject 1 (Sem 2): 35
⚠ Failed in Python (Marks: 35.0), enter new marks.
Re-enter marks for Python: 70
☑ Final mark for Python: 70.0
Enter name of Subject 2 (Sem 2): Linux
Enter marks for Linux: 80
☑ Final mark for Linux: 80.0
--- Semester 2 Summary ---
Total: 150.0
Average: 75.00
Grade: B
Enter number of subjects for Semester 3: 2
Enter name of Subject 1 (Sem 3): Switching
Enter marks for Switching: 67
☑ Final mark for Switching: 67.0

09.04.25