TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

LAB 2

By:

Sinjal Dahal (081/BEL/080)

DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING

LALITPUR, NEPAL

1. Write a program to input *n* numbers and store them in a list. Then perform the following operations:

i) Using built-in functions

ii) Without using built-in functions

a. Find the maximum and minimum number

b. Sort the list in ascending order

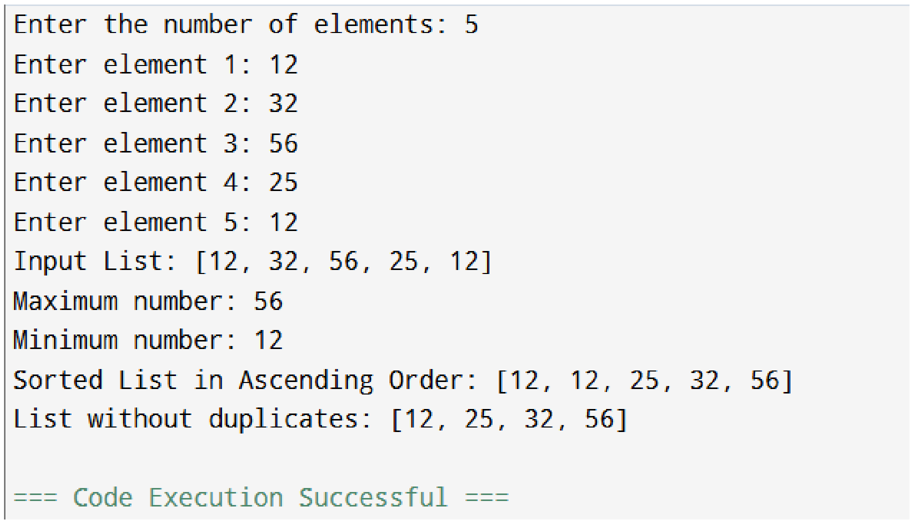
c. Remove duplicate elements

i) Using built-in functions

```
n = int(input("Enter the number of elements: "))
input_list = []

for i in range(n):
    val = int(input(f"Enter element {i + 1}: "))
    input_list.append(val)

print("Input List:", input_list)

max_num = max(input_list)
min_num = min(input_list)

print("Maximum number:", max_num)
print("Minimum number:", min_num)
```

```
sorted_list = sorted(input_list)
print("Sorted List in Ascending Order:", sorted_list)

unique_list = list(set(input_list))
unique_list.sort()
print("List without duplicates:", unique_list)
```

Output:



```
Enter the number of elements: 5
Enter element 1: 12
Enter element 2: 32
Enter element 3: 56
Enter element 4: 25
Enter element 5: 12
Input List: [12, 32, 56, 25, 12]
Maximum number: 56
Minimum number: 12
Sorted List in Ascending Order: [12, 12, 25, 32, 56]
List without duplicates: [12, 25, 32, 56]

=== Code Execution Successful ===
```

ii) Without using built-in functions

```
n = int(input("Enter the number of elements: "))
input_list = []

for i in range(n):
    val = int(input(f"Enter element {i + 1}: "))
    input_list.append(val)
```

```
print("Input List:", input_list)

max_num = input_list[0]
min_num = input_list[0]

for i in range(1, n):
    if input_list[i] > max_num:
        max_num = input_list[i]
    if input_list[i] < min_num:
        min_num = input_list[i]

print("Maximum number:", max_num)
print("Minimum number:", min_num)

sorted_list = input_list[:]
for i in range(n):
    for j in range(0, n - i - 1):
        if sorted_list[j] > sorted_list[j + 1]:
            # Swap
            temp = sorted_list[j]
            sorted_list[j] = sorted_list[j + 1]
            sorted_list[j + 1] = temp

print("Sorted List in Ascending Order:", sorted_list)
```

```
unique_list = []
for i in range(n):
    is_duplicate = False
    for j in range(len(unique_list)):
        if input_list[i] == unique_list[j]:
            is_duplicate = True
            break
    if not is_duplicate:
        unique_list.append(input_list[i])

print("List without duplicates:", unique_list)
```

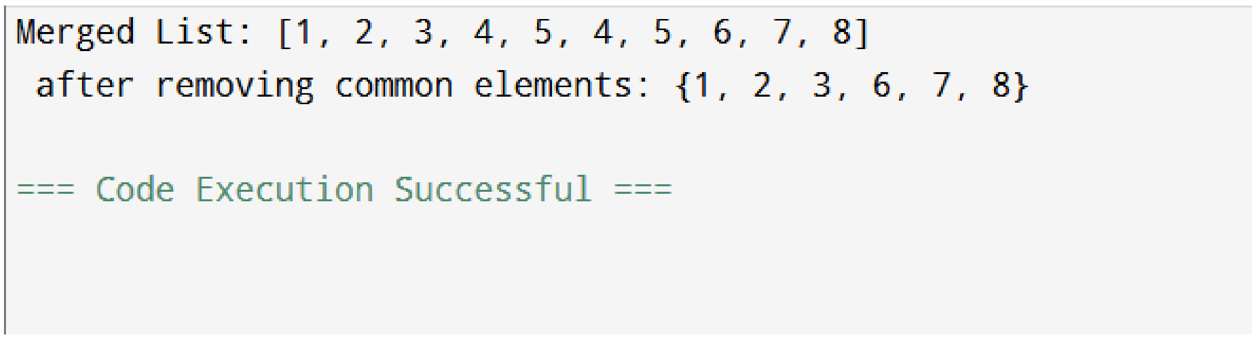
Output:

```
Enter the number of elements: 5
Enter element 1: 12
Enter element 2: 96
Enter element 3: 56
Enter element 4: 25
Enter element 5: 56
Input List: [12, 96, 56, 25, 56]
Maximum number: 96
Minimum number: 12
Sorted List in Ascending Order: [12, 25, 56, 56, 96]
List without duplicates: [12, 96, 56, 25]
```

2. Given two lists of integers, write a program to merge them into a single list and then remove the elements that are common in both.

```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]


list3 = list1 + list2


print("Merged List:", list3)


list3 = set(list3)


for i in list1:
    if i in list2:
        list3.remove(i)


print(" after removing common elements:", list3)
```

Output:



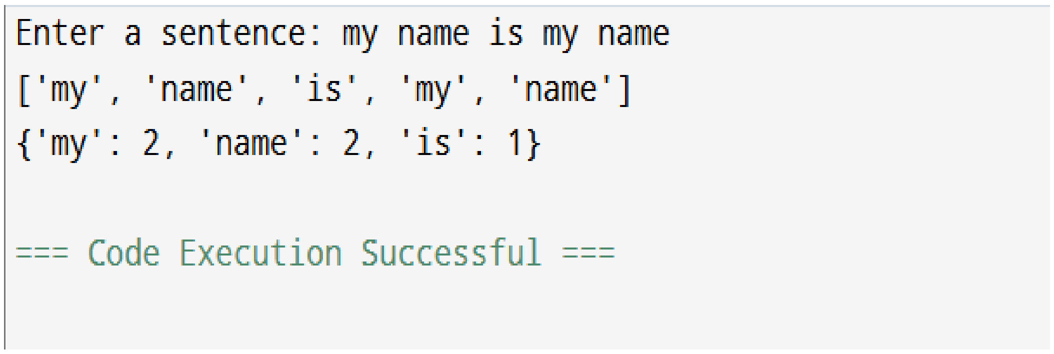
```
Merged List: [1, 2, 3, 4, 5, 4, 5, 6, 7, 8]
 after removing common elements: {1, 2, 3, 6, 7, 8}

=== Code Execution Successful ===
```

3. Create a program that reads a sentence from the user and stores each word as an element of a list. Then count the frequency of each word using only lists.

```
input_sentence = list(input("Enter a sentence: ").split())
```

```
print(input_sentence)
input_sentence_1 = set(input_sentence)
word_count = {}
for i in input_sentence_1:
    word_count[i]= input_sentence.count(i)
print(word_count)
```

Output:



```
Enter a sentence: my name is my name
['my', 'name', 'is', 'my', 'name']
{'my': 2, 'name': 2, 'is': 1}

=== Code Execution Successful ===
```

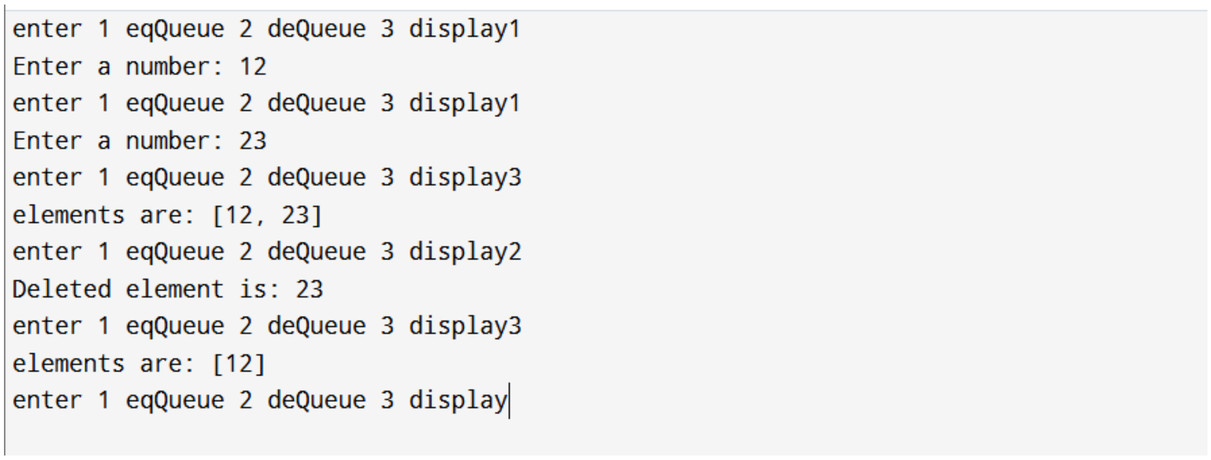
4. Write a program to simulate a basic stack and queue using a list. Provide options to:

* **Push**
* **Pop (stack)**
* **Enqueue**
* **Dequeue (queue)**

```
queue = []
while True:
    choice = input("enter 1 eqQueue 2 deQueue 3 display")
```

```
if choice=="1":
    num = int(input("Enter a number: "))
    queue.append(num)
elif choice=="2":
    if(len(queue)==0):
        print("queue is empty")
        continue
    print(f"Deleted element is: {queue.pop(1)}")
elif choice=="3":
    if(len(queue)==0):
        print("queue is empty")
        continue
    print(f"elements are: {queue}")
else:
    print("invalid choice")
    break
```

Output:

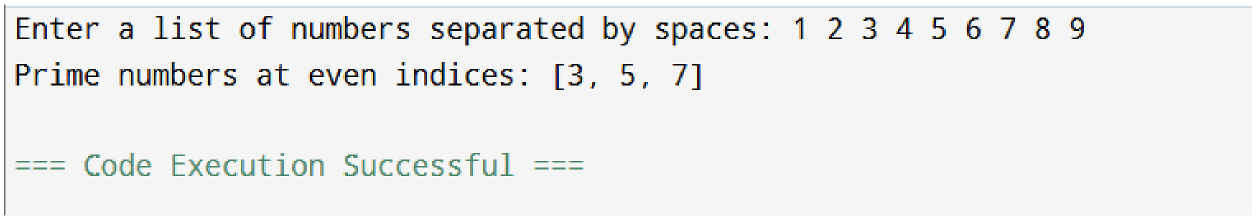


```
enter 1 eqQueue 2 deQueue 3 display1
Enter a number: 12
enter 1 eqQueue 2 deQueue 3 display1
Enter a number: 23
enter 1 eqQueue 2 deQueue 3 display3
elements are: [12, 23]
enter 1 eqQueue 2 deQueue 3 display2
Deleted element is: 23
enter 1 eqQueue 2 deQueue 3 display3
elements are: [12]
enter 1 eqQueue 2 deQueue 3 display
```

5. Write a Python function that accepts a list and returns a new list containing only the elements at even indexes and those that are prime numbers.

```
def is_prime(num):
    count = 0
    for i in range(1, num + 1):
        if num % i == 0:
            count += 1
    return count == 2

def primes_at_even_indices(lst):
    result = []
    even_indices = range(0, len(lst), 2)

    for index in even_indices:
        if is_prime(lst[index]):
            result.append(lst[index])
    return result

l = input("Enter a list of numbers separated by spaces: ").split()
l = [int(x) for x in l]

filtered = primes_at_even_indices(l)
print("Prime numbers at even indices:", filtered)
```

Output:



```
Enter a list of numbers separated by spaces: 1 2 3 4 5 6 7 8 9
Prime numbers at even indices: [3, 5, 7]

=== Code Execution Successful ===
```

6. Write a program to create a tuple of *n* numbers, then find:

a. The average of the numbers

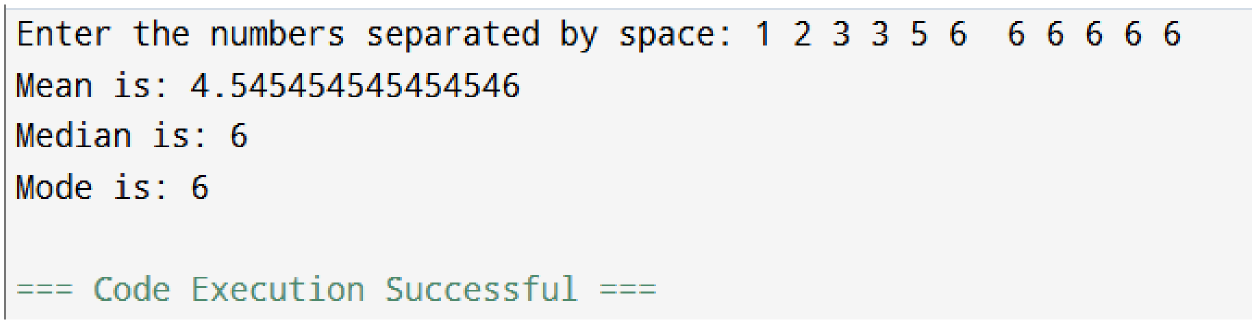
b. The median

c. The mode (without using libraries)

```
tuple_a = input("Enter the numbers separated by space: ").split()
tuple_1 = tuple(int(x) for x in tuple_a)


def average(tup):
    total = 0
    for i in tup:
        total += i
    return total / len(tup)


def median(tup):


    sorted_tup = list(tup)
    for i in range(len(sorted_tup)):
        for j in range(i + 1, len(sorted_tup)):
            if sorted_tup[i] > sorted_tup[j]:
                sorted_tup[i], sorted_tup[j] = sorted_tup[j], sorted_tup[i]

    n = len(sorted_tup)
    if n % 2 == 1:
        return sorted_tup[n // 2]
    else:
        mid1 = sorted_tup[n // 2 - 1]
        mid2 = sorted_tup[n // 2]
        return (mid1 + mid2) / 2

def mode(tup):
    max_count = 0
    result = tup[0]

    for i in tup:
        count = 0
        for j in tup:
            if i == j:
                count += 1
        if count > max_count:
            max_count = count
            result = i
    return result
```

```
print(f"Mean is: {average(tuple_1)}")

print(f"Median is: {median(tuple_1)}")

print(f"Mode is: {mode(tuple_1)}")
```

Output:

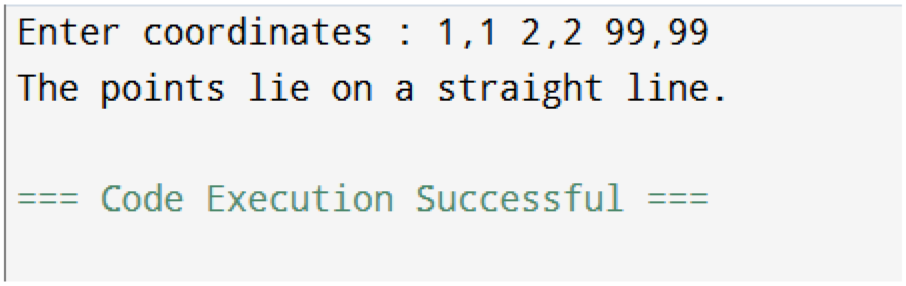


```
Enter the numbers separated by space: 1 2 3 3 5 6  6 6 6 6 6
Mean is: 4.545454545454546
Median is: 6
Mode is: 6

=== Code Execution Successful ===
```

7. Write a program that receives a list of tuples representing (x, y) coordinates. Determine whether the points form a straight line.

```
def is_straight_line(points):
    x1, y1 = points[0]
    x2, y2 = points[1]

    for i in range(2, len(points)):
        x3, y3 = points[i]
        if (y2 - y1) * (x3 - x2) != (y3 - y2) * (x2 - x1):
            return False
    return True

raw = input("Enter coordinates : ")
```

```
points = [eval(p) for p in raw.split()]


if is_straight_line(points):
    print("The points lie on a straight line.")
else:
    print("The points do NOT lie on a straight line.")
```



```
Enter coordinates : 1,1 2,2 99,99
The points lie on a straight line.

=== Code Execution Successful ===
```

8. Write a program to input two sets of student roll numbers: one who play cricket and another who play football. Find:

  a. Students who play both sports

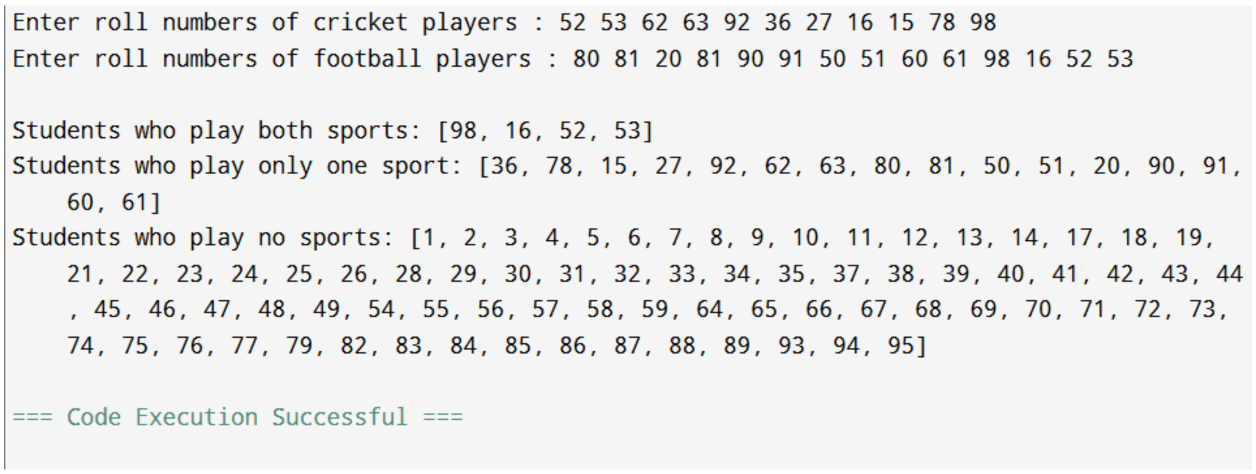
  b. Students who play only one sport

  c. Students who play neither (given a master list of all students)

```
cricket_input = input("Enter roll numbers of cricket players : ")
cricket = set(int(x) for x in cricket_input.split())


football_input = input("Enter roll numbers of football players : ")
football = set(int(x) for x in football_input.split())
```

```
master = set(range(1, 96))

def both_players(cricket, football):
    res = []
    for i in cricket:
        if i in football:
            res.append(i)
    return res

def only_one(cricket, football):
    res = []
    for i in cricket:
        if i not in football:
            res.append(i)
    for j in football:
        if j not in cricket:
            res.append(j)
    return res

def no_one(master, cricket, football):
    all_sports = cricket.union(football)
    res = []
    for i in master:
        if i not in all_sports:
            res.append(i)
    return res


print("\nStudents who play both sports:", both_players(cricket, football))

print("Students who play only one sport:", only_one(cricket, football))

print("Students who play no sports:", no_one(master, cricket, football))
```

Output:

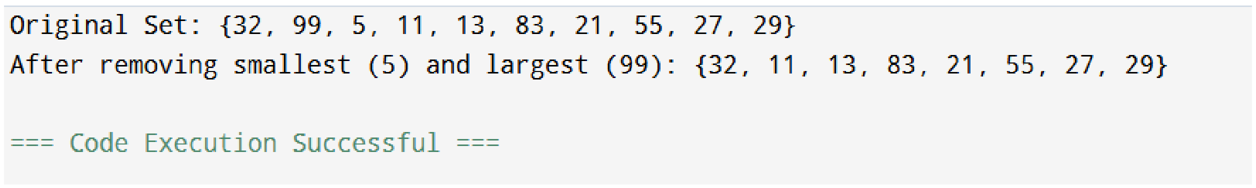


```
Enter roll numbers of cricket players : 52 53 62 63 92 36 27 16 15 78 98
Enter roll numbers of football players : 80 81 20 81 90 91 50 51 60 61 98 16 52 53

Students who play both sports: [98, 16, 52, 53]
Students who play only one sport: [36, 78, 15, 27, 92, 62, 63, 80, 81, 50, 51, 20, 90, 91,
    60, 61]
Students who play no sports: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 17, 18, 19,
    21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44
    , 45, 46, 47, 48, 49, 54, 55, 56, 57, 58, 59, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
    74, 75, 76, 77, 79, 82, 83, 84, 85, 86, 87, 88, 89, 93, 94, 95]

=== Code Execution Successful ===
```

9. Create a set of random numbers. Add more numbers until the set has 10 unique elements. Also, remove the smallest and largest element.

```
import random


def generate_set():

    nums = set()




    while len(nums) < 10:
        nums.add(random.randint(1, 100))

    print("Original Set:", nums)



    smallest = min(nums)

    largest = max(nums)



    nums.remove(smallest)

    nums.remove(largest)


    print(f"After removing smallest ({smallest}) and largest ({largest}):", nums)


generate_set()
```

Output:

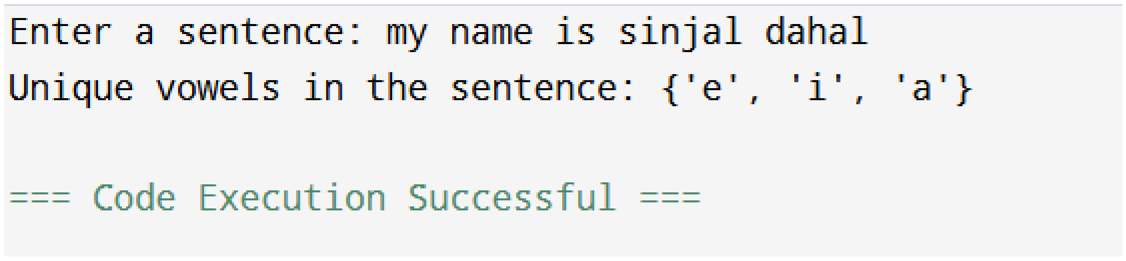


```
Original Set: {32, 99, 5, 11, 13, 83, 21, 55, 27, 29}
After removing smallest (5) and largest (99): {32, 11, 13, 83, 21, 55, 27, 29}

=== Code Execution Successful ===
```

10. Write a Python function that accepts a sentence and returns a set of all unique vowels used.

```
sentence = input("Enter a sentence: ")

vowels = 'aeiou'
unique_vowels = []

for ch in sentence.lower():
    if ch in vowels and ch not in unique_vowels:
        unique_vowels.append(ch)

print("Unique vowels in the sentence:", set(unique_vowels))
```

Output:



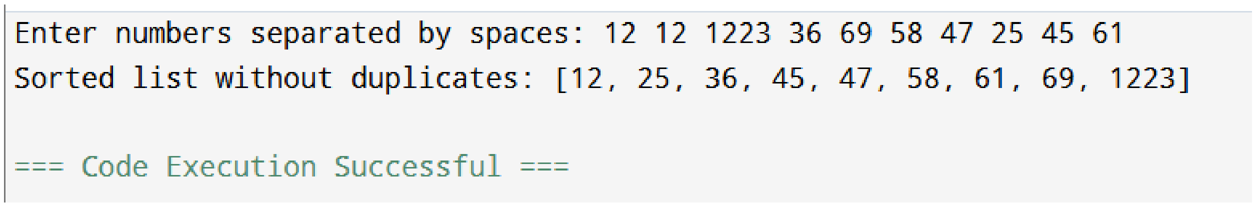
```
Enter a sentence: my name is sinjal dahal
Unique vowels in the sentence: {'e', 'i', 'a'}

=== Code Execution Successful ===
```

11. Given a list of numbers with duplicates, use a set to remove the duplicates. Then, convert it back to a sorted list and display the result.

```
numbers = input("Enter numbers separated by spaces: ").split()

numbers = [int(num) for num in numbers]
```

```
unique_sorted = sorted(set(numbers))


print("Sorted list without duplicates:", unique_sorted)
```

Output:



```
Enter numbers separated by spaces: 12 12 1223 36 69 58 47 25 45 61
Sorted list without duplicates: [12, 25, 36, 45, 47, 58, 61, 69, 1223]

=== Code Execution Successful ===
```

12. Create a dictionary to store student names as keys and their scores in three subjects as values (in a list). Write functions to:

a. Display the average marks of each student

b. Find the topper

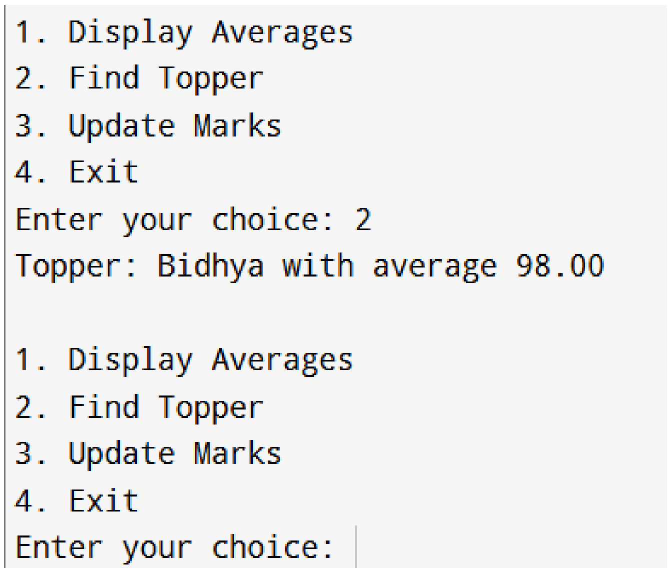
c. Update the marks of a student

```
students = {
    "Bidhya": [85, 90, 78],
    "Puspha": [92, 88, 95],
    "Sher": [70, 75, 80]
}

def display_averages(student_dict):
    for name, marks in student_dict.items():
        average = sum(marks) / len(marks)
        print(f"{name}: {average:.2f}")

def find_topper(student_dict):
    topper = ""
    highest = 0
    for name in student_dict:
        avg = sum(students[name]) / len(students[name])
        if avg > highest:
            highest = avg
            topper = name
    print(f"Topper: {topper} with average {highest:.2f}")

def update_marks(student_dict):
    name = input("Enter student name: ")
    marks = input("Enter 3 marks separated by space: ").split()
    marks = [int(m) for m in marks]
    student_dict[name] = marks
    print(f"{name}'s marks updated.")

while True:
    print("\n1. Display Averages")
    print("2. Find Topper")
    print("3. Update Marks")
    print("4. Exit")

    ch = input("Enter your choice: ")
```

```
if ch == '1':
    display_averages(students)
elif ch == '2':
    find_topper(students)
elif ch == '3':
    update_marks(students)
elif ch == '4':
    break
else:
    print("Invalid choice.")
```

Output:

```
1. Display Averages
2. Find Topper
3. Update Marks
4. Exit
Enter your choice: 1
Bidhya: 84.33
Puspha: 91.67
Sher: 75.00

1. Display Averages
2. Find Topper
3. Update Marks
4. Exit
Enter your choice: 3
Enter student name: Bidhya
Enter 3 marks separated by space: 99 98 97
Bidhya's marks updated.
```



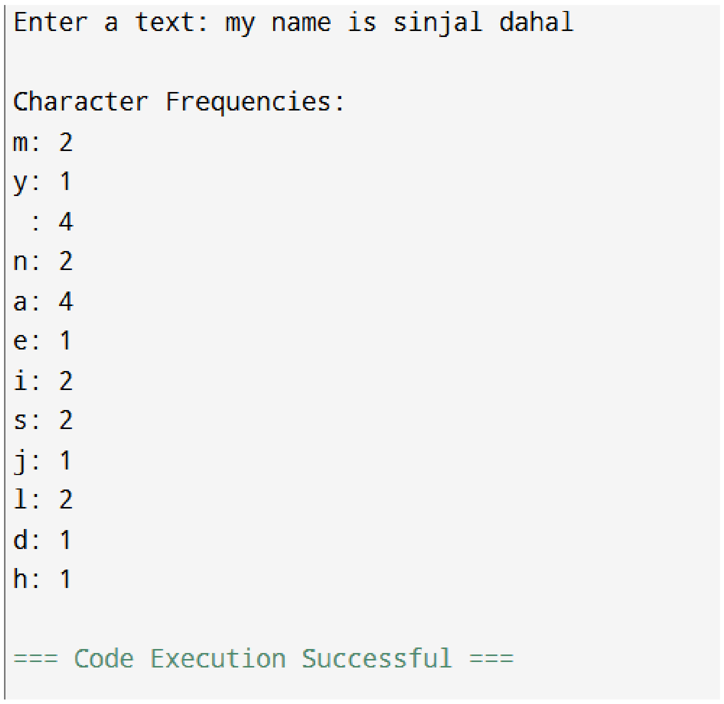
```
1. Display Averages
2. Find Topper
3. Update Marks
4. Exit
Enter your choice: 2
Topper: Bidhya with average 98.00

1. Display Averages
2. Find Topper
3. Update Marks
4. Exit
Enter your choice:
```

13. Write a program that reads a text and counts the frequency of each character (excluding spaces and special characters) using a dictionary.

```
text = input("Enter a text: ")


frequency = {}


for char in text:


    char = char.lower()
    if char in frequency:
        frequency[char] += 1
    else:
        frequency[char] = 1


print("\nCharacter Frequencies:")
```

```
for char, count in frequency.items():
    print(f"{char}: {count}")
```

Output:



```
Enter a text: my name is sinjal dahal

Character Frequencies:
m: 2
y: 1
 : 4
n: 2
a: 4
e: 1
i: 2
s: 2
j: 1
l: 2
d: 1
h: 1

=== Code Execution Successful ===
```

14. Build a dictionary where the keys are product names and the values are their prices. Implement options to:

a. Add a new product

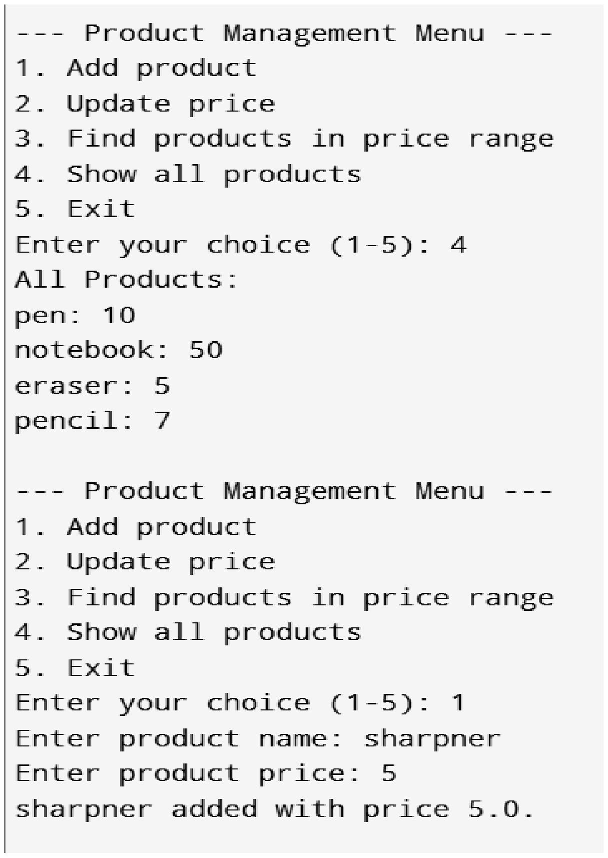
b. Update price of an existing product

c. Find products within a given price range

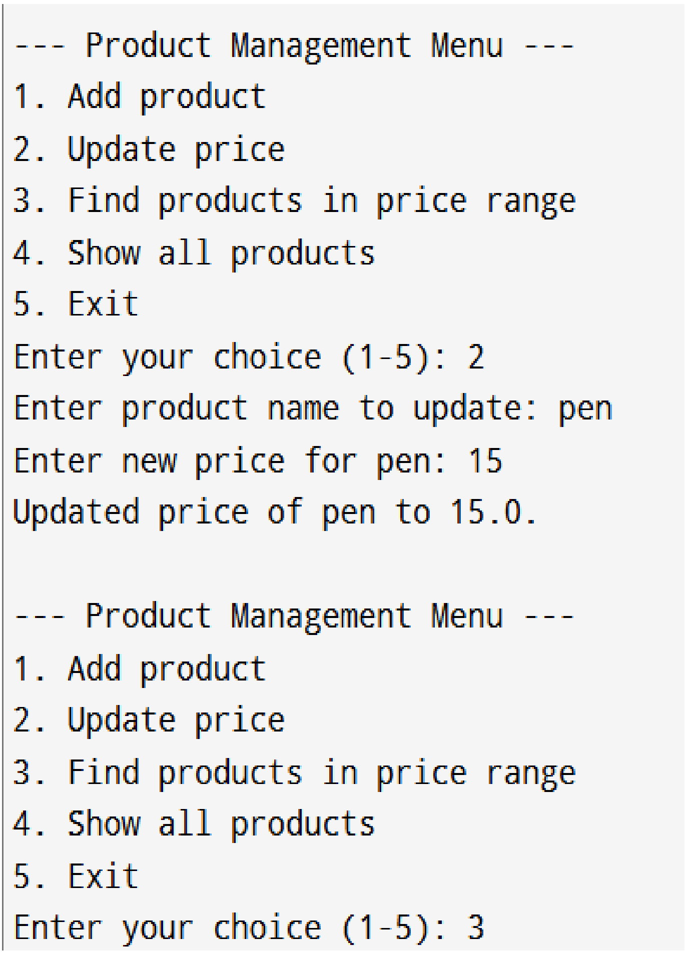
```
products = {
    "pen": 10,
    "notebook": 50,
    "eraser": 5,
    "pencil": 7
}

def add_product():
    name = input("Enter product name: ").lower()
    if name in products:
        print("Product already exists.")
    else:
        price = float(input("Enter product price: "))
        products[name] = price
        print(f"{name} added with price {price}.")

def update_price():
    name = input("Enter product name to update: ").lower()
    if name in products:
        new_price = float(input(f"Enter new price for {name}: "))
        products[name] = new_price
        print(f"Updated price of {name} to {new_price}.")
    else:
        print("Product not found.")

def find_in_range():
    low = float(input("Enter minimum price: "))
    high = float(input("Enter maximum price: "))
    found = False
    print("Products in range:")
    for name, price in products.items():
        if low <= price <= high:
            print(f"{name}: {price}")
            found = True
    if not found:
        print("No products found in this range.")

while True:
    print("\n--- Product Management Menu ---")
    print("1. Add product")
    print("2. Update price")
    print("3. Find products in price range")
    print("4. Show all products")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == '1':
        add_product()
    elif choice == '2':
        update_price()
    elif choice == '3':
        find_in_range()
```

```
elif choice == '4':
    print("All Products:")
    for k, v in products.items():
        print(f"{k}: {v}")
elif choice == '5':
    print("Exiting.")
    break
else:
    print("Invalid choice.")
```

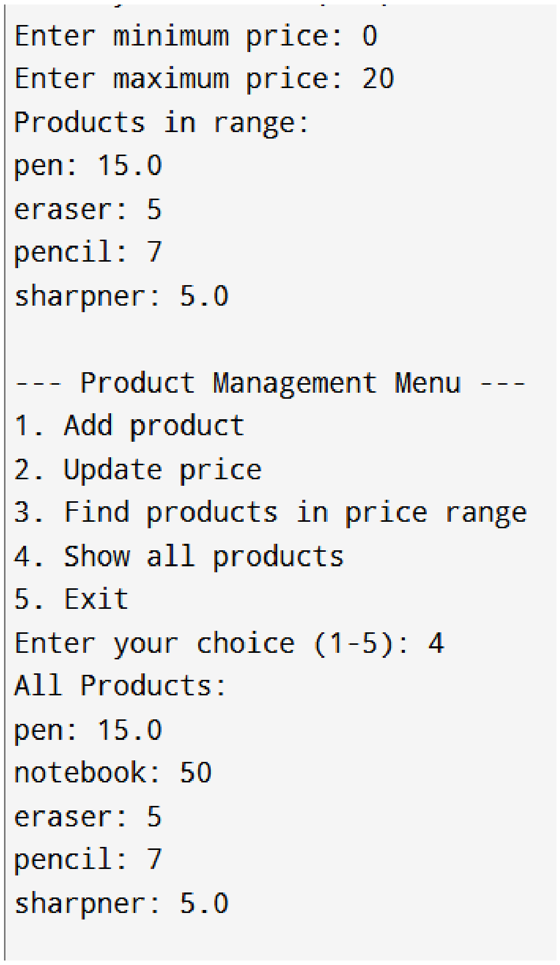
Output:



```
--- Product Management Menu ---
1. Add product
2. Update price
3. Find products in price range
4. Show all products
5. Exit
Enter your choice (1-5): 4
All Products:
pen: 10
notebook: 50
eraser: 5
pencil: 7

--- Product Management Menu ---
1. Add product
2. Update price
3. Find products in price range
4. Show all products
5. Exit
Enter your choice (1-5): 1
Enter product name: sharpner
Enter product price: 5
sharpner added with price 5.0.
```



```
--- Product Management Menu ---
1. Add product
2. Update price
3. Find products in price range
4. Show all products
5. Exit
Enter your choice (1-5): 2
Enter product name to update: pen
Enter new price for pen: 15
Updated price of pen to 15.0.

--- Product Management Menu ---
1. Add product
2. Update price
3. Find products in price range
4. Show all products
5. Exit
Enter your choice (1-5): 3
```



```
Enter minimum price: 0
Enter maximum price: 20
Products in range:
pen: 15.0
eraser: 5
pencil: 7
sharpner: 5.0

--- Product Management Menu ---
1. Add product
2. Update price
3. Find products in price range
4. Show all products
5. Exit
Enter your choice (1-5): 4
All Products:
pen: 15.0
notebook: 50
eraser: 5
pencil: 7
sharpner: 5.0
```

### MINI PROJECT: Student Report Card Management System

Problem Statement:

Design and implement a Student Report Card Management System using Python that allows a teacher to:

*** Add new student records (name, roll number, subject-wise marks)**

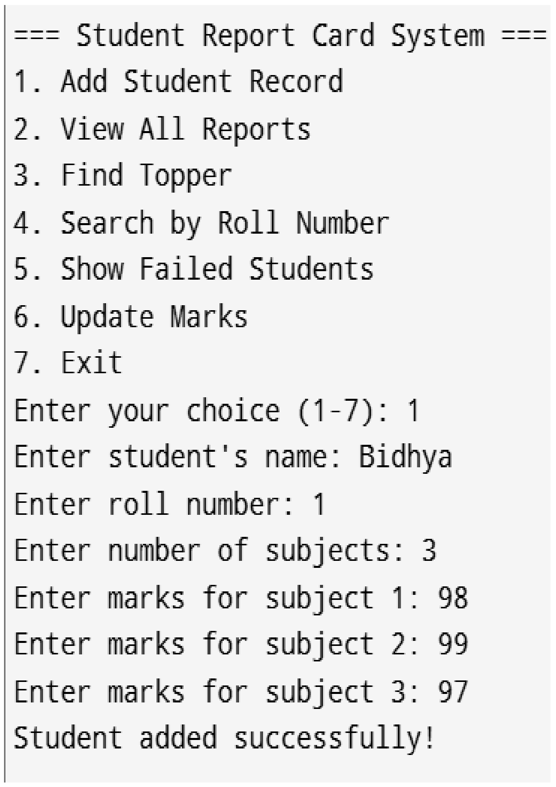
*** View the report of all students**

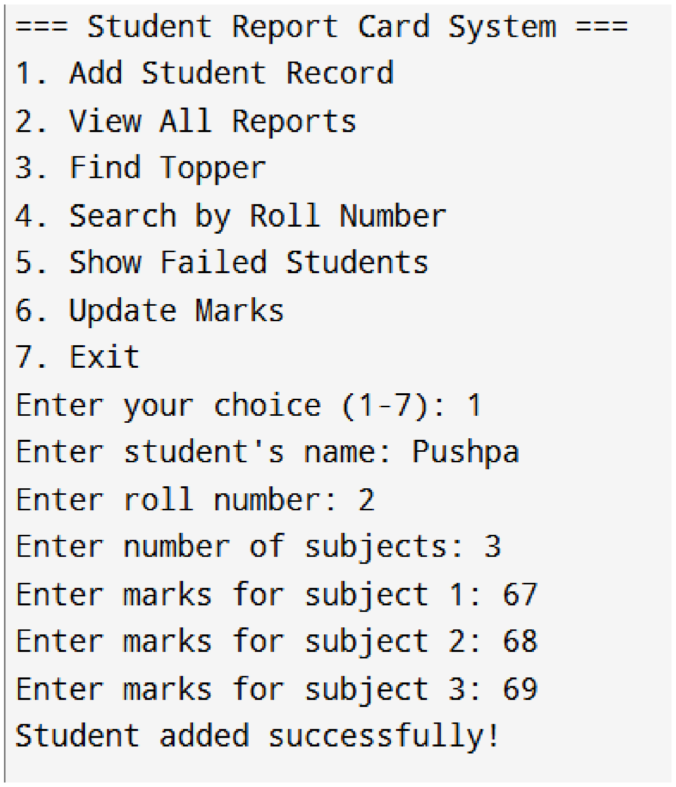
*** Display the topper(s) of the class based on average marks**

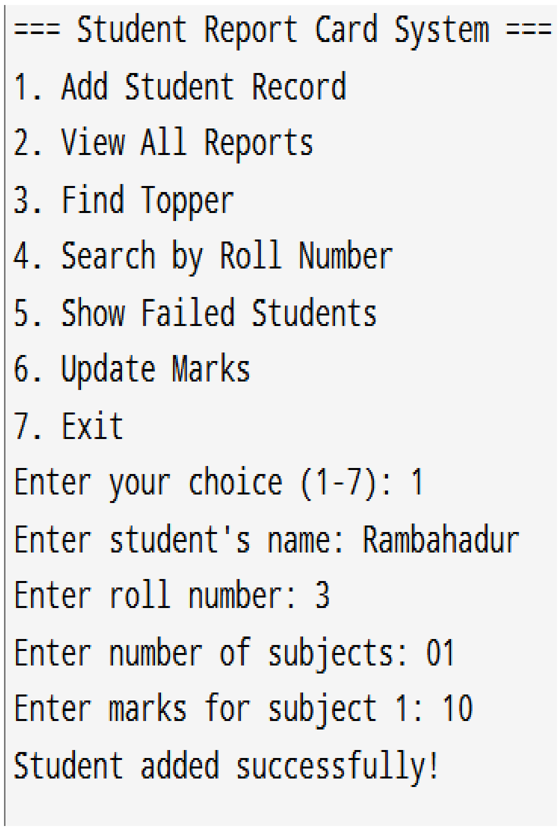
* **Search for a student by roll number**
* **Display all students who have failed in one or more subjects**
* **Optionally update marks of any student**

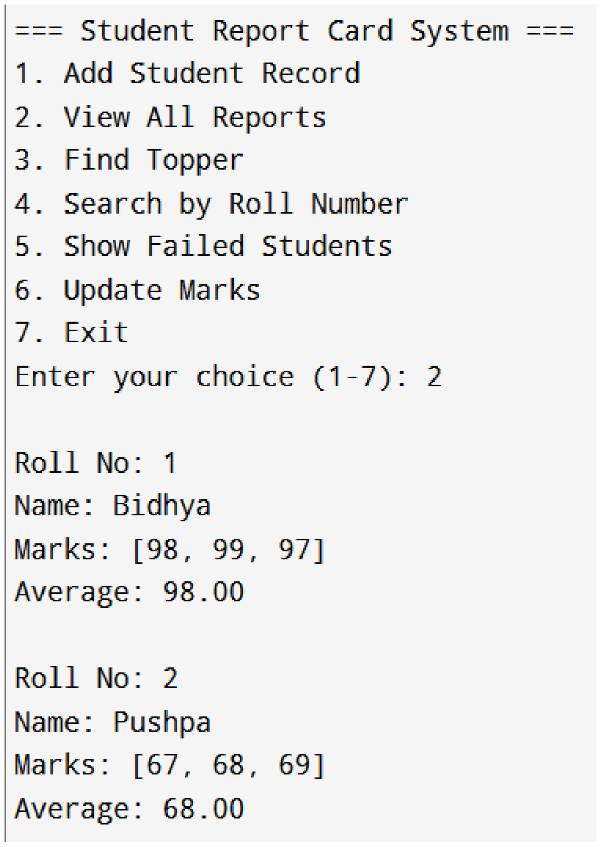
```
students = {}
def add_student():
    name = input("Enter student's name: ")
    roll = input("Enter roll number: ")
    marks = []
    subjects = int(input("Enter number of subjects: "))
    for i in range(subjects):
        mark = int(input(f"Enter marks for subject {i+1}: "))
        marks.append(mark)
    students[roll] = {
        "name": name,
        "marks": marks
    }
    print("Student added successfully!")


def view_all():
    if not students:
        print("No student records found.")
        return
    for roll, data in students.items():
        print(f"\nRoll No: {roll}")
        print(f"Name: {data['name']}")
        print(f"Marks: {data['marks']}")
        avg = sum(data["marks"]) / len(data["marks"])
        print(f"Average: {avg:.2f}")


def find_topper():
    if not students:
        print("No student records to evaluate.")
        return

    max_avg = -1
    toppers = []

    for roll, data in students.items():
        avg = sum(data["marks"]) / len(data["marks"])
        if avg > max_avg:
            max_avg = avg
            toppers = [data["name"]]
        elif avg == max_avg:
            toppers.append(data["name"])

    print(f"Topper with average {max_avg:.2f}: {', '.join(toppers)}")


def search_student():
    roll = input("Enter roll number to search: ")
    if roll in students:
        data = students[roll]
        print(f"Name: {data['name']}")
        print(f"Marks: {data['marks']}")
        avg = sum(data["marks"]) / len(data["marks"])
        print(f"Average: {avg:.2f}")
    else:
        print("Student not found.")


def failed_students():
    print("\nStudents who failed in one or more subjects:")
    found = False
    for roll, data in students.items():
        if any(m < 35 for m in data["marks"]):
            print(f"Roll: {roll}, Name: {data['name']}, Marks: {data['marks']}")
            found = True
    if not found:
        print("No student has failed.")


def update_marks():
    roll = input("Enter roll number of student to update marks: ")
    if roll in students:
        new_marks = []
        subjects = len(students[roll]["marks"])
        for i in range(subjects):
            mark = int(input(f"Enter new marks for subject {i+1}: "))
            new_marks.append(mark)
        students[roll]["marks"] = new_marks
        print("Marks updated successfully.")
    else:
        print("Student not found.")


while True:
    print("\n=== Student Report Card System ===")
    print("1. Add Student Record")
    print("2. View All Reports")
    print("3. Find Topper")
    print("4. Search by Roll Number")
    print("5. Show Failed Students")
    print("6. Update Marks")
    print("7. Exit")

    choice = input("Enter your choice (1-7): ")

    if choice == '1':
        add_student()
    elif choice == '2':
        view_all()
    elif choice == '3':
        find_topper()
```

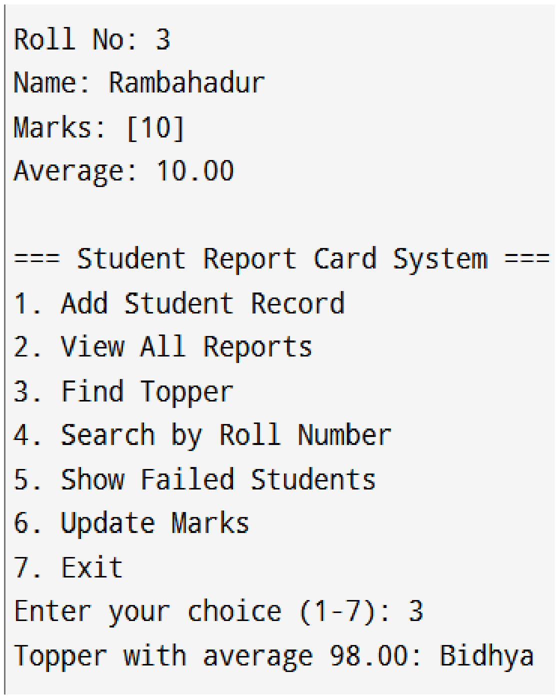
```
elif choice == '4':
    search_student()
elif choice == '5':
    failed_students()
elif choice == '6':
    update_marks()
elif choice == '7':
    print("Exiting program.")
    break
else:
    print("Invalid choice. Try again.")
```


```
=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 1
Enter student's name: Bidhya
Enter roll number: 1
Enter number of subjects: 3
Enter marks for subject 1: 98
Enter marks for subject 2: 99
Enter marks for subject 3: 97
Student added successfully!
```

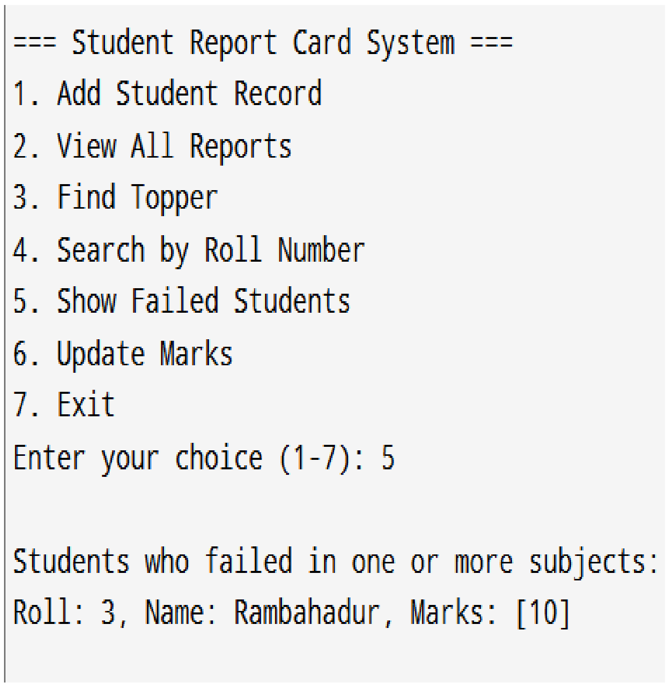

```
=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 1
Enter student's name: Pushpa
Enter roll number: 2
Enter number of subjects: 3
Enter marks for subject 1: 67
Enter marks for subject 2: 68
Enter marks for subject 3: 69
Student added successfully!
```


```
=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 1
Enter student's name: Rambahadur
Enter roll number: 3
Enter number of subjects: 01
Enter marks for subject 1: 10
Student added successfully!
```

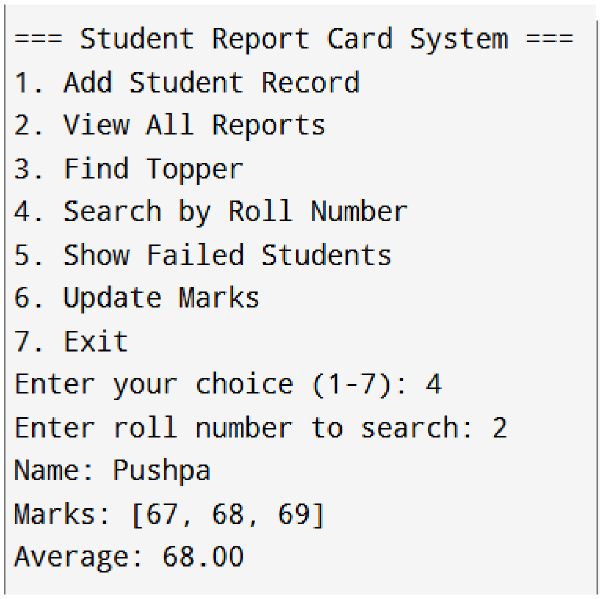

```
=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 2

Roll No: 1
Name: Bidhya
Marks: [98, 99, 97]
Average: 98.00

Roll No: 2
Name: Pushpa
Marks: [67, 68, 69]
Average: 68.00
```



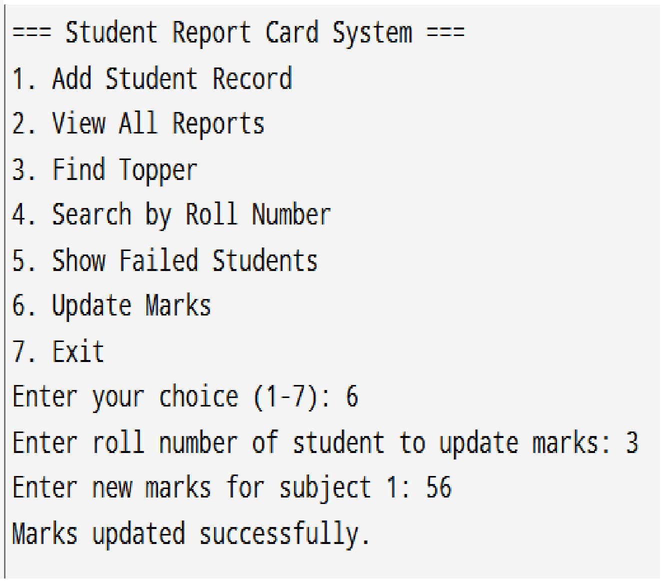
```
Roll No: 3
Name: Rambahadur
Marks: [10]
Average: 10.00

=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 3
Topper with average 98.00: Bidhya
```



```
=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 5

Students who failed in one or more subjects:
Roll: 3, Name: Rambahadur, Marks: [10]
```



```
=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 4
Enter roll number to search: 2
Name: Pushpa
Marks: [67, 68, 69]
Average: 68.00
```



```
=== Student Report Card System ===
1. Add Student Record
2. View All Reports
3. Find Topper
4. Search by Roll Number
5. Show Failed Students
6. Update Marks
7. Exit
Enter your choice (1-7): 6
Enter roll number of student to update marks: 3
Enter new marks for subject 1: 56
Marks updated successfully.
```

SINJAL DAHAL

081BEL080

https://github.com/sinjaldahal/sinjaldahal_BEL/tree/main/LabWork/lab_2