

DSL Assignment 1

Code:

```
print("*****Welcome!*****")
marks_list = [67, '', 99, 100, 'AB', 68, 70, 90, 68, 83, 'NA', 55, 76, 60, 88]
print('The list of marks scored by students in subject FDS are as follows:')
print(marks_list)
```

```
def menu():
    print('1. The average score of the class.')
    print('2. Highest score and lowest score of the class.')
    print('3. Count of students who were absent for the test.')
    print('4. Marks with highest frequency.')
    print('5. Close menu.')
    choice = int(
        input('Enter appropriate number to execute the following task:'))

    if choice == 1:
        print('**** Average score of the class ****')
        averageScore()
        print("*****")
        menu()

    elif choice == 2:
        print('**** Highest and lowest score of the class ****')
        highestScore()
        lowestScore()
        print("*****")
        menu()
```

```

elif choice == 3:

    print("**** Count of students who were absent ****")

    absentCount()

    print("*****")

    menu()

elif choice == 4:

    print("**** Marks with highest frequency ****")

    freqHigh()

    print("*****")

    menu()

elif choice == 5:

    exit

else:

    print("Bro, I can only handle 5 things. Read the menu again!")

    menu()

```

Definition of functions used above

```

def averageScore():

    count = avg = total = 0

    n = len(marks_list)

    print('Strength of class is: ', n)

    for i in marks_list:

        if type(i) == type(" "):      # Don't use isalpha() here.

```

```
        # Here, count is the no of students who were absent.
        count += 1
    else:
        total = total+i
    avg = total/(n-count)
    print('Average is: ', avg)
```

```
def highestScore():
    max = count = 0
    for i in marks_list:
        if type(i) == type(" "):      # Don't use isalpha() here.
            count += 1
        elif i > max:
            max = i
    print('Highest Score achieved is: ', max)
```

```
def lowestScore():
    # assumed min value to be max and will change it by comparing with list of marks
    min = 100
    count = 0
    for i in marks_list:
        if type(i) == type(" "):
            count += 1
        elif i < min:
            min = i
    print("Lowest Score achieved is: ", min)
```

```
def absentCount():
    count = 0
```

```
for i in marks_list:
    if type(i) == type(" "):
        count += 1
print("No. of absent students: ", count)
```

```
def freqHigh():
    max = 0
    res = marks_list[0]
    for i in marks_list:
        freq = marks_list.count(i)
        if freq > max:
            max = freq
            res = i
    print('Highest Frequency: ', max)
    print('Marks which has highest frequency:', str(res))
```

```
menu()
```

Output:

*****Welcome!*****

The list of marks scored by students in subject FDS are as follows:

[67, '', 99, 100, 'AB', 68, 70, 90, 68, 83, 'NA', 55, 76, 60, 88]

1. The average score of the class.
2. Highest score and lowest score of the class.
3. Count of students who were absent for the test.
4. Marks with highest frequency.
5. Close menu.

Enter appropriate number to execute the following task: 1

**** Average score of the class ****

Strength of class is: 15

Average is: 77.0

Enter appropriate number to execute the following task:2

**** Highest and lowest score of the class ****

Highest Score achieved is: 100

Lowest Score achieved is: 55

Enter appropriate number to execute the following task:3

**** Count of students who were absent ****

No. of absent students: 3

Enter appropriate number to execute the following task:4

**** Marks with highest frequency ****

Highest Frequency: 2

Marks which has highest frequency: 68

DSL Assignment 2

Code:

```
def LongestWord():
```

```
    str = []
```

```
    string = input("Enter the string: ")
```

```
    str = string.split()
```

```
    max_len = len(str[0])
```

```
    temp = str[0]
```

```
    for word in str:
```

```
        if(len(word) > max_len):
```

```
            max_len = len(word)
```

```
            temp = word
```

```
    print("The word with maximum length is: ", temp)
```

```
    print("its length = ", max_len)
```

```
def Frequency():
```

```
    str = input("Enter some string: ")
```

```
    dict = {}
```

```
    for i in str:
```

```
        keys = dict.keys()
```

```
        if i in keys:
```

```
            dict[i] += 1
```

```
        else:
```

```
            dict[i] = 1
```

```
    chr = input("Enter the character to get its frequency: ")
```

```
    if (chr in str):
```

```
    print(f"The frequency of character {chr} is: ", end="")
    print(dict[chr])
else:
    print(f"{chr} is not present.")
    print(dict[chr])
```

```
def Palindrome():
    str = input("\n Enter some string: ")
    if(str == str[::-1]):
        print("\n The given string is palindrome")
    else:
        print("\n The given string is not palindrome")
```

```
def Find_Substr():
    str = input("\n Enter some statement: ")
    word = input("\n Enter the substring to be searched: ")
    for i in range(len(str) - len(word)+1):
        if (str[i:i+len(word)] == word):
            return i
    return 'Not Found'
```

```
def OccurWords():
    str = input("\n Enter some statement: ")
    counts = dict()
    words = str.split()
    for word in words:
        if word in counts:
            counts[word] += 1
    else:
```

```
        counts[word] = 1
    print(counts)

print("***** Program for String operations *****")
while(True):
    print("\n 1. To display word with the longest length")
    print("\n 2. To determine the frequency of occurrence of particular character in the string")
    print("\n 3. To check whether given string is palindrome or not")
    print("\n 4. To display index of first appearance of the substring")
    print("\n 5. To count the occurrences of each word in a given string")
    print("\n Enter your choice: ", end="")

    choice = int(input())
    if(choice == 1):
        LongestWord()
    elif(choice == 2):
        Frequency()
    elif(choice == 3):
        Palindrome()
    elif(choice == 4):
        print(" The given substring is found at index: ", Find_Substr())
    elif(choice == 5):
        OccurWords()
    else:
        print("Exiting")
        break
```


Output:

***** Program for String operations *****

1. To display word with the longest length
2. To determine the frequency of occurrence of particular character in the string
3. To check whether given string is palindrome or not
4. To display index of first appearance of the substring
5. To count the occurrences of each word in a given string

Enter your choice: 1

Enter the string: classroom college university

The word with maximum length is: university

its length = 10

Enter your choice: 2

Enter some string: occurrence

Enter the character to get its frequency: r

The frequency of character r is: 2

Enter your choice: 3

Enter some string: man

The given string is not palindrome

Enter your choice: 4

Enter some statement: the beautiful garden

Enter the substring to be searched: beau

The given substring is found at index: 4

Enter your choice: 5

Enter some statement: man man tan tan

{'man': 2, 'tan': 2}

DSL Assignment 3

Code:

```
def addition(A, B):
```

```
    result = [[A[i][j] + B[i][j]
```

```
               for j in range(len(A[0]))] for i in range(len(A))]
```

```
    print("The Addition of Two Matrices...")
```

```
    for r in result:
```

```
        print(r)
```

```
def subtraction(A, B):
```

```
    result = [[A[i][j] - B[i][j]
```

```
               for j in range(len(A[0]))] for i in range(len(A))]
```

```
    print("The Subtraction of Two Matrices...")
```

```
    for r in result:
```

```
        print(r)
```

```
def transpose(A):
```

```
    result = [[0 for col in range(col_num)] for row in range(row_num)]
```

```
    for i in range(len(A)):
```

```
        for j in range(len(A[0])):
```

```
            result[j][i] = A[i][j]
```

```
    print("Transposed Matrix is ...")
```

```
    for r in result:
```

```
        print(r)
```

```

def multiplication(A, B):
    result = [[0 for col in range(col_num)] for row in range(row_num)]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                result[i][j] += A[i][k] * B[k][j]

    print("Matrix Multiplication is ...")
    for r in result:
        print(r)

row_num = int(input("Input number of rows: "))
col_num = int(input("Input number of columns: "))
A = [[0 for col in range(col_num)] for row in range(row_num)]
for row in range(row_num):
    for col in range(col_num):
        item = int(input("Enter the elements in first matrix: "))
        A[row][col] = item

print("The first matrix is...")
print(A)

B = [[0 for col in range(col_num)] for row in range(row_num)]
for row in range(row_num):
    for col in range(col_num):
        item = int(input("Enter the elements in second matrix: "))
        B[row][col] = item

```

```
print("The second matrix is...")
```

```
print(B)
```

```
print("\n Program for Matrix Operations")
```

```
while(True):
```

```
    print("\n 1. Addition of Two Matrices")
```

```
    print("\n 2. Subtraction of Two Matrices")
```

```
    print("\n 3. Multiplication of Two Matrices")
```

```
    print("\n 4. Transpose of Matrix")
```

```
    print("\n 5. Exit ")
```

```
    print("\n Enter your choice: ")
```

```
    choice = int(input())
```

```
    if(choice == 1):
```

```
        addition(A, B)
```

```
    elif(choice == 2):
```

```
        subtraction(A, B)
```

```
    elif(choice == 3):
```

```
        multiplication(A, B)
```

```
    elif(choice == 4):
```

```
        transpose(A)
```

```
    else:
```

```
        print("\n Exiting")
```

```
        break
```

Output:

Input number of rows: 2

Input number of columns: 2

Enter the elements in first matrix: 20

Enter the elements in first matrix: 30

Enter the elements in first matrix: 40

Enter the elements in first matrix: 55

The first matrix is...

[[20, 30], [40, 55]]

Enter the elements in second matrix: 20

Enter the elements in second matrix: 12

Enter the elements in second matrix: 2

Enter the elements in second matrix: 66

The second matrix is...

[[20, 12], [2, 66]]

Program for Matrix Operations

1. Addition of Two Matrices
2. Subtraction of Two Matrices
3. Multiplication of Two Matrices
4. Transpose of Matrix
5. Exit

Enter your choice: 1

The Addition of Two Matrices...

[40, 42]

[42, 121]

Enter your choice: 2

The Subtraction of Two Matrices...

[0, 18]

[38, -11]

Enter your choice: 3

Matrix Multiplication is ...

[460, 2220]

[910, 4110]

Enter your choice: 4

Transposed Matrix is ...

[20, 40]

[30, 55]

DSL Assignment 4

Code:

try:

```
def Ternary_Search(min_index, max_index, search, lsort):

    if (max_index >= min_index):

        m1 = min_index + (max_index - min_index)//3
        m2 = max_index - (max_index - min_index)//3

        if (lsort[int(m1)] == search):
            return m1

        if (lsort[int(m2)] == search):
            return m2

        if (search < lsort[m1]):
            return Ternary_Search(min_index, m1-1, search, lsort)

        elif (search > lsort[m2]):
            return Ternary_Search(m2+1, max_index, search, lsort)

        else:
            return Ternary_Search(m1+1, m2-1, search, lsort)

    return -1
```

```
def Ternary_Non_Recursive(min_index, max_index, search, lsort):

    while max_index >= min_index:

        mid1 = min_index + (max_index-min_index) // 3
```



```
mid2 = max_index - (max_index-min_index) // 3
```

```
if search == lsort[mid1]:
```

```
    return mid1
```

```
if search == mid2:
```

```
    return mid2
```

```
if search < lsort[mid1]:
```

```
    r = mid1 - 1
```

```
elif search > lsort[mid2]:
```

```
    min_index = mid2 + 1
```

```
else:
```

```
    min_index = mid1 + 1
```

```
    max_index = mid2 - 1
```

```
return -1
```

```
def Sort(lsort):
```

```
    f = len(lsort)
```

```
    for i in range(f-1):
```

```
        for j in range(0, f-i-1):
```

```
            if lsort[j] > lsort[j + 1]:
```

```
                lsort[j], lsort[j + 1] = lsort[j + 1], lsort[j]
```

```
    return(lsort)
```

```

def accept():
    l = []
    n = int(input("\nEnter the total Number Of Students :"))
    print("-----")
    for i in range(0, n):
        try:
            m = float(input("Enter The Roll Number: "))

            l.append(int(m))
        except(ValueError):
            print("Invalid Input")
            m = float(input("Enter The Roll Number: "))
            l.append(int(m))
    return l

```

```

def menu(l):

    right = len(l)
    left = 0
    lsort = Sort(l)
    print("Sorted Roll Numbers are : ", lsort)

    print("-----")
    print("1.Ternary Search")
    print("2.Ternary Search (Non Recursive Approach)")
    print("3.Re-enter The Roll Number List")
    print("4.Exit")
    print("-----")
    opt = int(input("Enter The Action To Be Performed: "))

```

```

if opt == 1:
    search = int(
        input("Enter The Student Roll Number To Be Searched: "))

    res = Ternary_Search(left, right, search, lsort)
    if res != -1:
        print(
            f"Roll Number Found At Index {str(res)} In Sorted Roll Number List")
    else:
        print("Roll Number Is Not Present In List")
    choice = input("Continue ?(y/n): ")
    if(choice == "y"):
        print(
            "-----")
    ---\n\n\n")
        menu(l)
    if(choice == "n"):
        exit()

if opt == 2:
    search = int(
        input("Enter The Student Roll Number To Be Searched: "))

    res = Ternary_Non_Recursive(left, right, search, lsort)
    if res != -1:
        print(
            f"Roll Number Found At Index {str(res)} In Sorted Roll Number List")

    else:

```

```

        print("Roll Number Is Not Present In List")

    choice = input("Continue ?(y/n): ")
    if(choice == "y"):
        print(
            "-----
---\n\n\n")
        menu(l)

    if(choice == "n"):
        exit()
    if opt == 3:
        main()

    if opt == 4:
        choice = input("\n\nExit ?(y/n): ")
        if(choice == "y"):
            print("Exiting...")
            exit()

        if(choice == "n"):
            print(
                "-----
---\n\n\n")
            menu(l)

    else:
        print("Exiting...")
        exit()

```

```

def main():

    print("\n-----Ternary Search-----")
    l = accept()
    menu(l)

main()

except KeyboardInterrupt:

    choice = input("\n\nExit ?(y/n): ")

    if(choice == "y"):
        print("Exiting...")
        exit()

    if(choice == "n"):
        main()
    else:
        print("Exiting...")
        exit()

```

Output:

```

-----Ternary Search-----

```

Enter the total Number Of Students :5

Enter The Roll Number: 20

Enter The Roll Number: 65

Enter The Roll Number: 88

Enter The Roll Number: 45

Enter The Roll Number: 78

Sorted Roll Numbers are : [20, 45, 65, 78, 88]

1.Ternary Search

2.Ternary Search (Non Recursive Approach)

3.Re-enter The Roll Number List

4.Exit

Enter The Action to Be Performed: 1

Enter The Student Roll Number to Be Searched: 65

Roll Number Found at Index 2 In Sorted Roll Number List

Continue? (y/n): y

Sorted Roll Numbers are: [20, 45, 65, 78, 88]

1.Ternary Search

2.Ternary Search (Non-Recursive Approach)

3.Re-enter the Roll Number List

4.Exit

Enter The Action to Be Performed: 2

Enter The Student Roll Number to Be Searched: 45

Roll Number Found at Index 1 In Sorted Roll Number List

DSL Assignment 5

Code:

```
percentage_list = []

print("*****")

n = int(input("Enter the number of students: "))

print(f"Enter the percentage of {n} students: ")

for i in range(n):

    percentage = float(input())

    percentage_list.append(percentage)


def bubbleSort(percentage_list):

    n = len(percentage_list)

    for i in range(n):

        for j in range(0, n-i-1):

            if percentage_list[j] > percentage_list[j+1]:

                percentage_list[j], percentage_list[j +

                    1] = percentage_list[j+1], percentage_list[j]


print("Sorted list: ", end="")

print(percentage_list)


def selectionSort(itemsList):

    n = len(itemsList)

    for i in range(n - 1):

        minValueIndex = i

        for j in range(i + 1, n):

            if itemsList[j] < itemsList[minValueIndex]:

                minValueIndex = j
```

```

        if minValueIndex != i:
            temp = itemsList[i]
            itemsList[i] = itemsList[minValueIndex]
            itemsList[minValueIndex] = temp

    print("Sorted list: ", end="")
    print(percentage_list)

def topFiveScores(percentage_list):
    print("The top 5 scores are as follows: ")
    percentage_list = percentage_list[::-1]
    for i in range(5):
        print(percentage_list[i], end=" | ")

while True:
    print("\n*****")
    print("Which operation you would like to perform: ")
    print("1. Selection Sort")
    print("2. Bubble Sort")
    print("3. Exit")
    print("*****")
    selection = int(input("Enter your choice: "))
    if (selection == 1):
        selectionSort(percentage_list)
        print("Do you want to print the top 5 score!? Enter: yes/no ", end=" ")
        ch = input()
        if (ch == "yes"):
            topFiveScores(percentage_list)

```



```
elif (selection == 2):  
    bubbleSort(percentage_list)  
    print("Do you want to print the top 5 score!? yes/no", end=" ")  
    ch = input()  
    if (ch == "yes"):  
        topFiveScores(percentage_list)  
elif(selection == 3):  
    print("Exiting!")  
    break  
else:  
    print("Invalid choice.")
```

Output:

Enter the number of students: 5

Enter the percentage of 5 students:

45

98

55

68

78

Which operation you would like to perform:

1. Selection Sort

2. Bubble Sort

3. Exit

Enter your choice: 1

Sorted list: [45.0, 55.0, 68.0, 78.0, 98.0]

Do you want to print the top 5 score!? Enter: yes/no no

Enter your choice: 2

Sorted list: [45.0, 55.0, 68.0, 78.0, 98.0]

Do you want to print the top 5 score!? yes/no yes

The top 5 scores are as follows:

98.0 | 78.0 | 68.0 | 55.0 | 45.0 |

DSL Assignment 6

Code:

```
def partition(percentage_list, low, high):  
    i = (low-1)    # index of smaller element  
    pivot = percentage_list[high]    # pivot  
  
    for j in range(low, high):  
  
        # If current element is smaller than or  
        # equal to pivot  
        if percentage_list[j] <= pivot:  
  
            # increment index of smaller element  
            i = i+1  
            percentage_list[i], percentage_list[j] = percentage_list[j], percentage_list[i]  
  
    percentage_list[i+1], percentage_list[high] = percentage_list[high],  
percentage_list[i+1]  
    return (i+1)  
  
def quickSort(percentage_list, low, high):  
    if len(percentage_list) == 1:  
        return percentage_list  
    if low < high:  
  
        # pi is partitioning index, arr[p] is now  
        # at right place  
        pi = partition(percentage_list, low, high)
```

```
# Separately sort elements before
# partition and after partition
quickSort(percentage_list, low, pi-1)
quickSort(percentage_list, pi+1, high)
```

```
def displaySortedArray():
    print("Sorted array is:\n")
    for i in range(n):
        print(percentage_list[i], end=" | ")
    print("\n")
```

```
def topFiveScores(percentage_list):
    print("The top 5 scores are as follows:\n")
    percentage_list = percentage_list[::-1]
    for i in range(5):
        print(percentage_list[i], end=" | ")
    print("\n")
```

```
while True:
    print("\n\n*****")
    print("Which operation you would like to perform: ")
    print("1. Accept Data")
    print("2. Perform Quick Sort")
    print("3. Exit")
    print("*****")
    selection = int(input("Enter your choice: "))
    if (selection == 1):
        percentage_list = []
        print("*****")
```

```
n = int(input("Enter the number of students: "))
print(f"Enter the percentage of {n} students: ")
for i in range(n):
    percentage = float(input())
    percentage_list.append(percentage)

elif (selection == 2):
    quickSort(percentage_list, 0, (len(percentage_list)-1))
    displaySortedArray()
    print("\nDo you want to print the top 5 score!? Enter: yes/no ?", end=" ")
    ch = input()
    if (ch == "yes"):
        topFiveScores(percentage_list)

elif(selection == 3):
    print("Exiting!")
    break

else:
    print("Invalid choice.")
```

Output:

Which operation you would like to perform:

1. Accept Data
2. Perform Quick Sort
3. Exit

Enter your choice: 1

Enter the number of students: 5

Enter the percentage of 5 students:

78

99

65

45.5

55

Which operation you would like to perform:

1. Accept Data
2. Perform Quick Sort
3. Exit

Enter your choice: 2

Sorted array is:

45.5 | 55.0 | 65.0 | 78.0 | 99.0 |

Do you want to print the top 5 score!? Enter: yes/no ? yes

The top 5 scores are as follows:

99.0 | 78.0 | 65.0 | 55.0 | 45.5 |

DSL Assignment 7

Code:

```
#include <stdio.h>
#include <iostream>
#include <string>
using namespace std;
```

```
class list;
```

```
class node
```

```
{
    int prn;
    string name;
    node *next;
```

```
public:
```

```
    node(int
```

```
        x,
```

```
        string
```

```
        nm)
```

```
{
```

```
    prn = x;
```

```
    next = NULL;
```

```
    name = nm;
```

```
}
```

```
friend
```

```
    class list;
```

```
};
```

```
class list
```

```
{
    node *start;

public:
    list()
    {
        start = NULL;
    }
    void
    create();
    void
    display();
    void
    insertAtBeginning();
    void
    insertAtEnd();
    void
    insertAfter();
    void
    deleteAtFirst();
    void
    deleteByValue();
    void
    deleteAtEnd();
    int
    computeTotal();
    void
    sortList();
    void
```



```

concatList(list &q1);

void
displayRev(node *t);

bool
reverseDisplay() // function is only
// for passing start as argument to recursive function
{
    if (start == NULL)
        return false;

    node *temp = start;
    displayRev(temp);
    // cout << "(President)";

    return true;
}

};

void

list::displayRev(node *t)
{
    if (t == NULL)
        return;

    else
    {
        displayRev(t->next);

        cout << "\nPRN NO:" << t->prn << " Name: " << t->name;

    }
}

void list::create()
{

```

```

int
    no;
string
    nam;
if (start == NULL)
{
    cout << "Enter PRN number: ";

    cin >> no;

    cout << "Enter name: ";

    cin >> nam;

    start = new node(no, nam);

    cout << "\n===== List Created =====";
}
else
{
    cout << "\nList is already created.";
}
}

void list::display()
{
    node *t;

    t = start;

    if (start == NULL)
        cout << "\nList is Empty";

    else
    {
        cout << "\n===== List: =====\n";

        while (t != NULL)
        {

```

```

        cout << t->prn << " " << t->name << " \n";
        t = t->next;
    }
    // cout << t->prn << " " << t->name << " \n";
}
}
void

```

```

list::insertAtBeginning()
{
    int
        no;
    string
        nam;
    node *temp;
    if (start == NULL)
    {
        create();
    }
    else
    {
        cout << "\nEnter PRN Number : ";
        cin >> no;
        cout << "Enter Name : ";
        cin >> nam;
        // cout << nam;
        temp = new node(no, nam);
        temp->next = start;
        start = temp;
    }
}

```

```

        ;
        cout << "inserter" << temp->name << "in the beginning";
    }
}

void list::insertAtEnd()
{
    int
        no;

    string
        nam;

    node *t;
    if (start == NULL)
        create();
    else
    {
        cout << "\nEnter PRN Number : ";
        cin >> no;
        cout << "Enter Name : ";
        cin >> nam;

        t = start;
        while (t->next != NULL)
            t = t->next;

        node *p = new node(no, nam);
        t->next = p;
    }
}

void list::insertAfter()
{

```

```

int
    prev_no;
cout << "\nEnter PRN No. after do you want insert : ";
cin >> prev_no;
node *t;
t = start;
string
    nam;
int
    flag = 0,
    no;
while (t != NULL)
{
    if (t->prn == prev_no)
    {
        flag = 1;
        break;
    }
    t = t->next;
}
if (flag == 1)
{
    node *p;
    cout << "\nEnter PRN Number : ";
    cin >> no;
    cout << "Enter Name : ";
    cin >> nam;
    p = new node(no, nam);
    p->next = t->next;
}

```

```

        t->next = p;
    }
    else
    {
        cout << "\n"
              << prev_no << " is not in list.";
    }
}

void list::
    deleteAtFirst()
{
    node *t;
    if (start == NULL)
        cout << "\nClub is Empty..";
    else
    {
        t = start;
        start = start->next;
        t->next = NULL; // Not necessary

        delete t;
        cout << "\nPresident deleted..";
    }
}

void list::
    deleteByValue()
{
    int
        no,

```

```

    flag = 0;
node *t, *prev;
if (start == NULL)
    cout << "\nList/Club is empty";
else
{
    cout << "\nEnter PRN No. of member to be deleted : ";
    cin >> no;

    t = start->next; // t=start if we have to delete president also..start->next is first
member
    while (t->next != NULL)
    {
        if (t->prn == no)
        {
            flag = 1;
            break;
        }
        prev = t;
        t = t->next;
    }
    if (flag == 1)
    {
        prev->next = t->next;
        t->next = NULL;
        delete t;
        cout << "\nMember with PRN No: " << no << " is deleted.";
    }
    else
        cout << "\nMember not found in List./President or Secretary cannot be
deleted.";
}

```

```

    }
}

void list::
    deleteAtEnd()
{
    node *t, *prev;

    t = start;

    if (start == NULL)
        cout << "\nClub is Empty..";
    else
    {

        while (t->next != NULL)
        {
            prev = t;
            t = t->next;
        }

        prev->next = NULL;

        delete t;

        cout << "\nSecretary Deleted.";
    }
}

int list::computeTotal()
{
    node *t;

    int
        count = 0;

    t = start;

    if (start == NULL)

```



```

{
    cout << "\nList is empty.";
    return 0;
}

while (t != NULL)
{
    count++;
    t = t->next;
}

return count;
}

void list::sortList()
{
    node *i, *j, *last = NULL;

    int
        tprn;

    string
        tname;

    if (start == NULL)
    {
        cout << "\nList is empty.";
        return;
    }

    for (i = start; i->next != NULL; i = i->next)
    {
        for (j = start; j->next != last; j = j->next)
        {

            if ((j->prn) > (j->next->prn))

```

```

        {
            tprn = j->prn;
            tname = j->name;
            j->prn = j->next->prn;
            j->name = j->next->name;
            j->next->prn = tprn;
            j->next->name = tname;
        }
    }
}

cout << "\n List is sorted.";

display();
}

void list::concatList(list &q1)
{

    node *t, *p;
    t = q1.start;
    if (t == NULL)
    {

        cout << "\nList 2 is empty";

        return;
    }
    p = start; // first
    // list

```

```

while (p->next != NULL)
{
    p = p->next;
}

p->next = t;
q1.start = NULL; // second
                // list is set
                //      to
                //      null

cout
    << "\nAfter concatenation list : \n";

display();
}
int main()
{
    list *l;
    int
        choice,
        selectList;
    list
        l1,
        l2;
    l = &l1;
X:
    cout << "\nSelect List\n1.List 1\n2.List 2\nEnter choice : ";

    cin >> selectList;

```

```

if (selectList == 1)
{
    l = &l1;
}
else if (selectList == 2)
{
    l = &l2;
}
else
{

    cout << "\nWrong list Number.";
    goto X;
}
do
{

    cout << "\n1. Create\n2. Insert President\n3. Insert secretary\n4. Insert after
position(member)\n";

    cout << "5. Display list\n6. Delete President\n7.Delete Secretary\n8. Delete
Member\n9. Find total No. of members\n10. Sort list\n11. Reselect List";

    cout << "\n12. Combine lists\n13.Reverse Display\n14. Exit\nEnter your choice:
";

    cin >> choice;

    switch (choice)

    {

        case 1:

            l->create();

```

break;

case 2:

l->insertAtBeginning();

break;

case 3:

l->insertAtEnd();

break;

case

4:

l->insertAfter();

break;

case 5:

l->display();

break;

case 6:

l->deleteAtFirst();

break;

case

7:

l->deleteAtEnd();

break;

```

case

    8:

        l->deleteByValue();

        break;
case 9:

    cout << "\nTotal members(including President & Secretary) : " << l-
>computeTotal();

    break;
case 10:

    l->sortList();

    break;
case 11:

    goto X;

    break;
case 12:

    l1.concatList(l2);

    break;
case 13:

    l->reverseDisplay();

    break;
default:

    cout << "Wrong choice";

}

} while (choice != 0);

cout << "\n===== GOOD BYE =====\n";

return 0;

}

```

Output:

Select List

1.List 1

2.List 2

Enter choice: 1

1. Create

2. Insert President

3. Insert secretary

4. Insert after position(member)

5. Display list

6. Delete President

7.Delete Secretary

8. Delete Member

9. Find total No. of members

10. Sort list

11. Reselect List

12. Combine lists

13.Reverse Display

14. Exit

Enter your choice: 1

Enter PRN number: 101

Enter name: abc

===== List Created =====

Enter your choice: 2

Enter PRN Number : 102

Enter Name : def

Enter your choice: 4

Enter PRN No. after do you want insert : 102

Enter PRN Number : 104

Enter Name : efg

Enter your choice: 11

Select List

1.List 1

2.List 2

Enter choice: 2

1. Create

2. Insert President

3. Insert secretary

4. Insert after position(member)

5. Display list

6. Delete President

7.Delete Secretary

8. Delete Member

9. Find total No. of members

10. Sort list

11. Reselect List

12. Combine lists

13.Reverse Display

14. Exit

Enter your choice: 1

Enter PRN number: 201

Enter name: thf

===== List Created =====

Enter your choice: 3

Enter PRN Number : 203

Enter Name : yhj

Enter your choice: 12

After concatenation list :

===== List: =====

102 def

104 efg

101 abc

201 thf

203 yhj