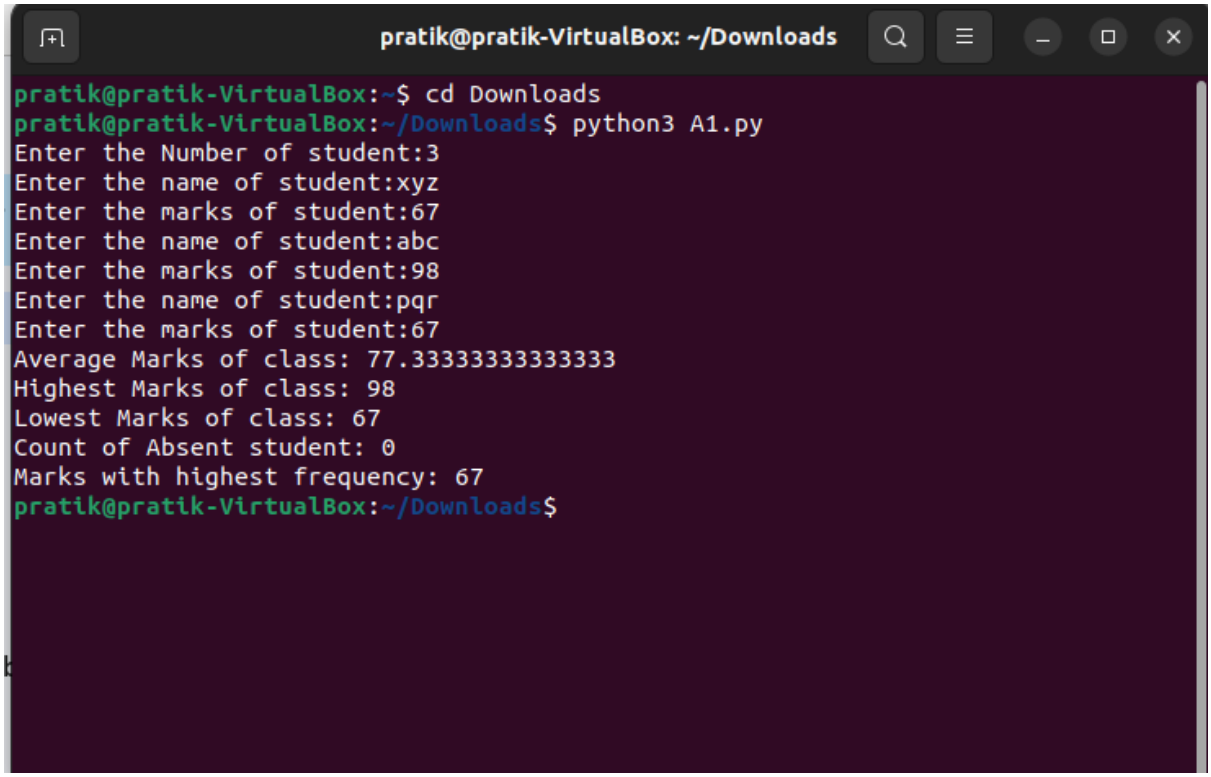
DSL Practical

Practical No

```
'''Write a Python program to store marks scored in subject "Fundamental
of Data Structure" by N students in the class.
Write functions to compute following:
a) The average score of class
b) Highest score and lowest score of class
c) Count of students who were absent for the test
d) Display mark with highest frequency'''



def avg(average):
    avg = sum(average) / len(average)
    print("Average Marks of class:", avg)

def maximum(average):
    maximum = max(average)
    print("Highest Marks of class:", maximum)

def minimum(average):
    minimum = min(average)
    print("Lowest Marks of class:", minimum)

def absent(marks_1, n):
    j = 0
    for k in range(n):
        if marks_1[k] < 0:
            j = j + 1
    print("Count of Absent student:", j)

def freq(frequency, n, marks_1):
    for j in range(n):
        f_1 = marks_1.count(marks_1[j])
        frequency.append(f_1)
    n_2 = frequency.index(max(frequency))
    if max(frequency) == 1:
        return ""
    else:
        return marks_1[n_2]

name_1 = []
marks_1 = []
average = []
frequency = []
n = int(input("Enter the Number of students:"))
for i in range(n):
    name = input("Enter the name of student:")
    marks = int(input("Enter the marks of student:"))
    if marks > 0:
        average.append(marks)
    name_1.append(name)
    marks_1.append(marks)

avg(average)
maximum(average)
```

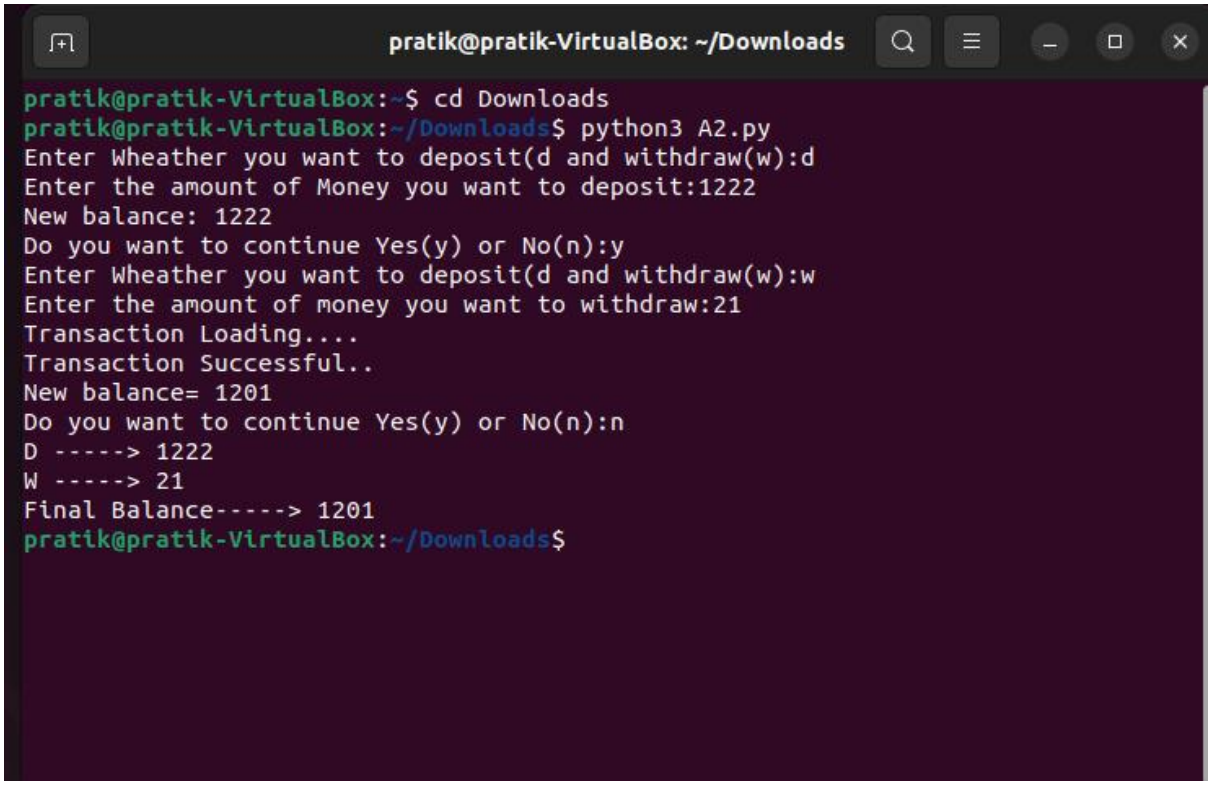
```
minimum(average)
absent(marks_1, n)
s
result = freq(frequency, n, marks_1)
if result:
    print("Marks with the highest frequency:", result)
```



Practical No 2

```
'''Write a Python program that computes the net amount of a bank
account based a transaction log from console input.
The transaction log format is shown as following:
D 100 W 200 (Withdrawal is not allowed if balance is going negative.
Write functions for withdraw and deposit) D means deposit while W means
withdrawal.
Suppose the following input is supplied to the program:
D 300, D 300 , W 200, D 100 Then, the output should be: 500'''
balance=0
i=1
l1=[]
l2=[]
l3=[]
while i==1:
    function=input("Enter Wheather you want to deposit(d and
withdraw(w):")
    function_1=function.capitalize()
    l1.append(function_1)
    if function_1=='D':
        d_amount=int(input("Enter the amount of Money you want to
deposit:"))
        if d_amount>0:
            balance=balance+d_amount
            print("New balance:",balance)
            l2.append(d_amount)
        else:
            print('Enter Valid Amount!!')
    elif function_1=='W':
        w_amount=int(input("Enter the amount of money you want to
withdraw:"))
        if w_amount>0:
            if w_amount<=balance:
```

```
                balance=balance-w_amount
                print("Transaction Loading....")
                print("Transaction Successful..\nNew balance=",balance)
                l2.append(w_amount)
            else:
                print("Getting Account Info....")
                print("Entered Amount is less than balance.")
                print("Transaction Unsucessful!!")
        else:
            print('Enter Valid Amount!!!')
    else:
        print('Invalid Input\nPlease enter only W or D:')
    cont=input("Do you want to continue Yes(y) or No(n):")
    cont_1=cont.capitalize()
    if cont_1=='Y':
        i=1
    else:
        i=i+1
l3=[l1,l2]
for i in range(len(l1)):
    print(l3[0][i],'----->',l3[1][i])
print("Final Balance----->",balance)
```

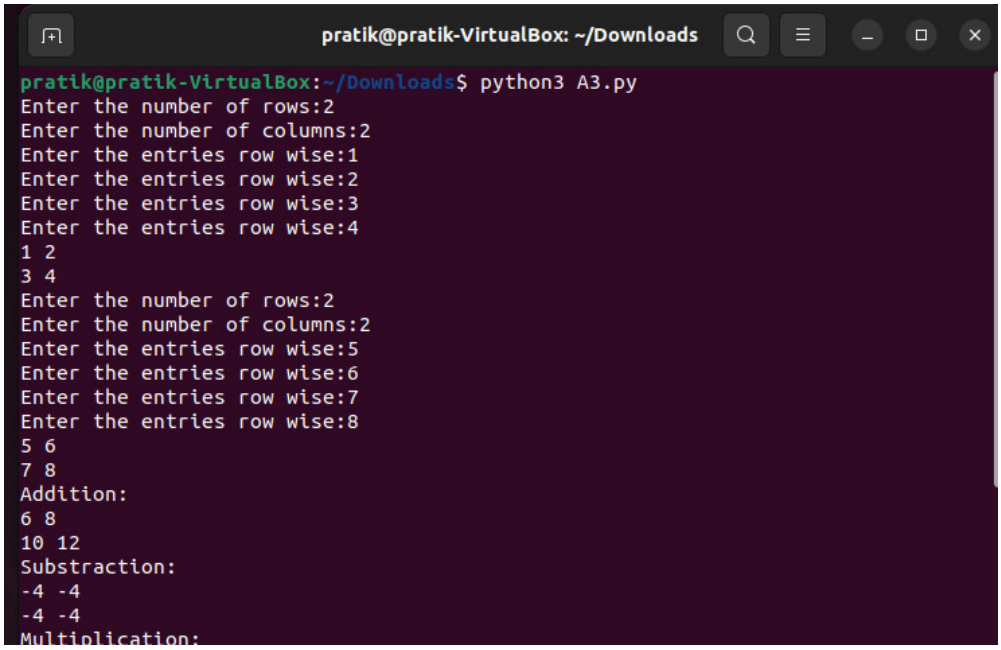


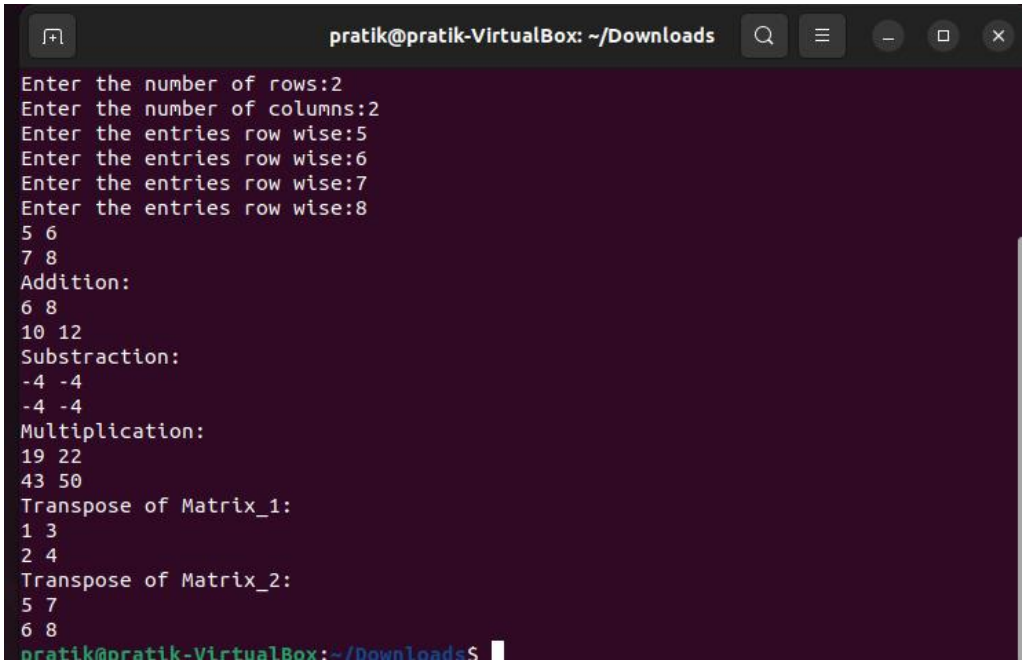
Practical No 3

```
'''Write a Python program to compute following computation on matrix:
a) Addition of two matrices b) Subtraction of two matrices
c) Multiplication of two matrices d) Transpose of a matrix'''
#Matrix 1
matrix_1=[]
r_1=int(input("Enter the number of rows:"))
c_1=int(input("Enter the number of columns:"))
for i_1 in range(r_1):
    a=[]
    for j_1 in range(c_1):
        n_1=int(input("Enter the entries row wise:"))
        a.append(n_1)
    matrix_1.append(a)
for l_1 in range(r_1):
    for k_1 in range(c_1):
        print(matrix_1[l_1][k_1],end=' ')
    print()


#Matrix 2

matrix_2=[]
r_2=int(input("Enter the number of rows:"))
c_2=int(input("Enter the number of columns:"))

for i_2 in range(r_2):
    a_2=[]
    for j_2 in range(c_2):
        n_2=int(input("Enter the entries row wise:"))
        a_2.append(n_2)
    matrix_2.append(a_2)

for l_2 in range(r_2):
    for k_2 in range(c_2):
```

```
        print(matrix_2[l_2][k_2],end=' ')
    print()

#Addition of Matrix

matrix_3=[]

if r_1==r_2 and c_1==c_2:
    for i_3 in range(r_1):
        a_3=[]
        for j_3 in range(c_1):
            n_3=matrix_1[i_3][j_3]+matrix_2[i_3][j_3]
            a_3.append(n_3)
        matrix_3.append(a_3)
    print("Addition:")
    for l_3 in range(r_1):
        for k_3 in range(c_1):
            print(matrix_3[l_3][k_3],end=' ')
        print()
else:
    print("Number of Rows and Columns of matrix should be same")
#Substraction of Matrix
if r_1==r_2 and c_1==c_2:
    matrix_4=[]
    for i_4 in range(r_1):
        a_4=[]
        for j_4 in range(c_1):
            n_4=matrix_1[i_4][j_4]-matrix_2[i_4][j_4]
            a_4.append(n_4)
        matrix_4.append(a_4)
    print("Substraction:")
    for l_4 in range(r_1):
        for k_4 in range(c_1):
            print(matrix_4[l_4][k_4],end=' ')
        print()
else:
    print("Number of Rows and Columns of matrix should be same")

#Multiplication of Matrix

matrix_5=[]
r_5=r_1
c_5=c_2

for i_5 in range(r_5):
    a_5=[]
    for j_5 in range(c_5):
        n_5 = 0
        for k_5 in range(c_5):
            n_5=n_5+matrix_1[i_5][k_5]*matrix_2[k_5][j_5]
        a_5.append(n_5)
    matrix_5.append(a_5)

print("Multiplication:")
for l_5 in range(r_5):
    for k_5 in range(c_5):
        print(matrix_5[l_5][k_5],end=' ')
    print()
```

```
#Transpose Of Matrix 1
matrix_6=[]
for i_6 in range(c_1):
    a_6=[]
    temp=0
    for j_6 in range(r_1):
        n_6=matrix_1[j_6][i_6]
        a_6.append(n_6)
    matrix_6.append(a_6)
print("Transpose of Matrix_1:")
for l_6 in range(c_1):
    for k_6 in range(r_1):
        print(matrix_6[l_6][k_6],end=' ')
    print()

#Transpose of Matrix 2
matrix_7=[]
for i_7 in range(c_2):
    a_7=[]
    temp=0
    for j_7 in range(r_2):
        n_7=matrix_2[j_7][i_7]
        a_7.append(n_7)
    matrix_7.append(a_7)
print("Transpose of Matrix_2:")
for l_7 in range(c_2):
    for k_7 in range(r_2):
        print(matrix_7[l_7][k_7],end=' ')
    print()
```



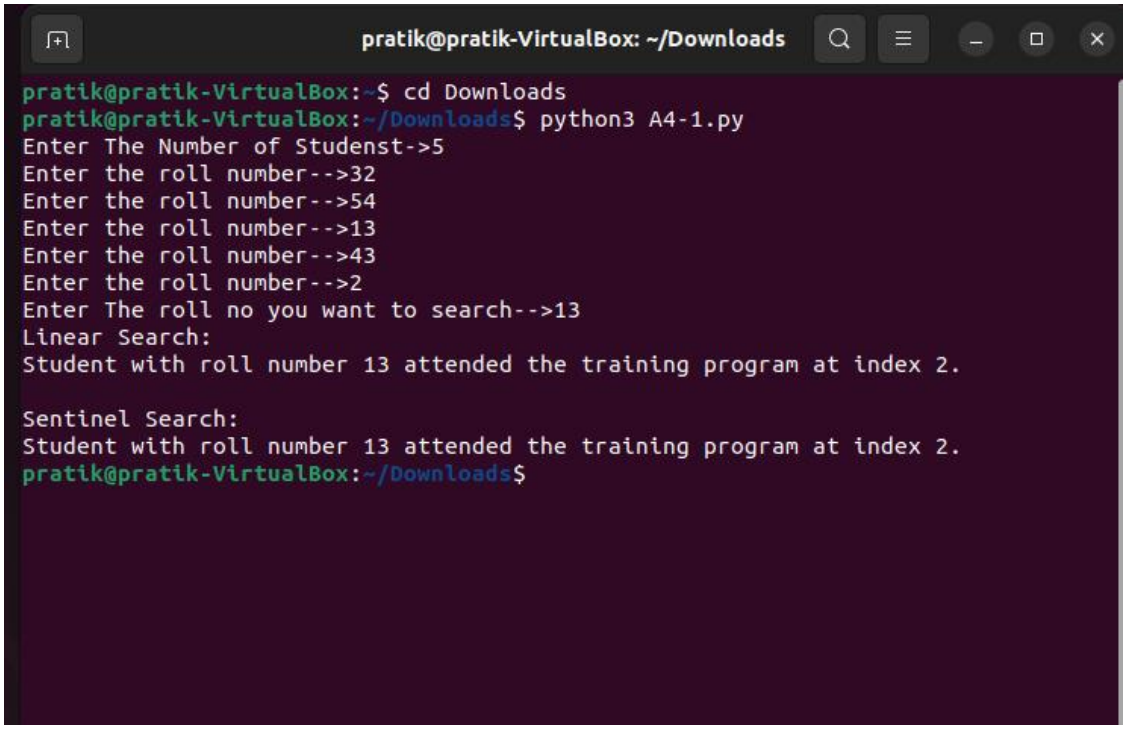


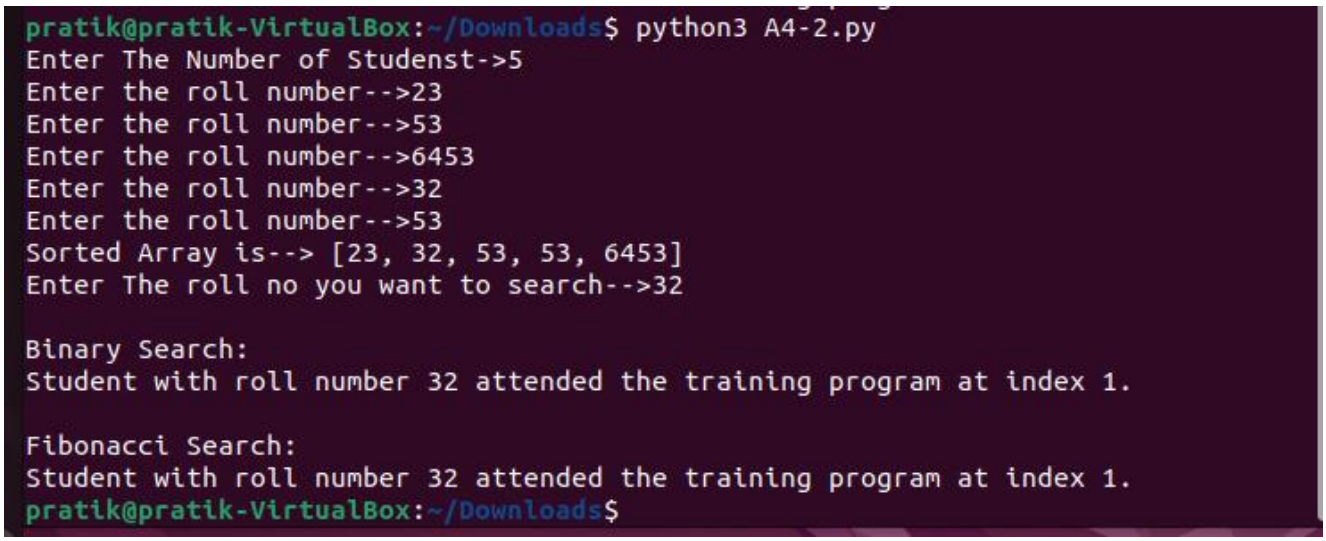
Practical No 4

```
def linear_search(arr, x):
    for i in range(len(arr)):
        if arr[i] == x:
            return True, i
    return False, -1

def sentinel_search(arr, x):
    last_element = arr[-1]
    arr[-1] = x

    i = 0
    while arr[i] != x:
        i += 1

    arr[-1] = last_element

    if i < len(arr) - 1 or arr[-1] == x:
        return True, i
    else:
        return False, -1

n=int(input("Enter The Number of Studenst->"))
r=[]
for i in range(n):
    student_roll = int(input("Enter the roll number to search-->"))
    r.append(student_roll)
student_roll=int(input("Enter The roll no you want to search-->"))
linear_result, linear_index = linear_search(r, student_roll)
sentinel_result, sentinel_index = sentinel_search(r, student_roll)
```

```
print("Linear Search:")
if linear_result:
    print(f"Student with roll number {student_roll} attended the
training program at index {linear_index}.")
else:
    print(f"Student with roll number {student_roll} did not attend the
training program.")

print("\nSentinel Search:")
if sentinel_result:
    print(f"Student with roll number {student_roll} attended the
training program at index {sentinel_index}.")
else:
    print(f"Student with roll number {student_roll} did not attend the
training program.")
def binary_search(arr, x):
    low, high = 0, len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        mid_value = arr[mid]

        if mid_value == x:
            return True, mid
        elif mid_value < x:
            low = mid + 1
        else:
            high = mid - 1

    return False, -1

def fibonacci_search(arr, x):
    fib_m_minus_2 = 0
    fib_m_minus_1 = 1
    fib = fib_m_minus_1 + fib_m_minus_2

    while fib < len(arr):
        fib_m_minus_2 = fib_m_minus_1
        fib_m_minus_1 = fib
        fib = fib_m_minus_1 + fib_m_minus_2

    offset = -1

    while fib > 1:
        i = min(offset + fib_m_minus_2, len(arr) - 1)

        if arr[i] < x:
            fib = fib_m_minus_1
            fib_m_minus_1 = fib_m_minus_2
            fib_m_minus_2 = fib - fib_m_minus_1
            offset = i

        elif arr[i] > x:
            fib = fib_m_minus_2
            fib_m_minus_1 = fib_m_minus_1 - fib_m_minus_2
            fib_m_minus_2 = fib - fib_m_minus_1

        else:
```

```
            return True, i

    if fib_m_minus_1 and arr[offset + 1] == x:
        return True, offset + 1

    return False, -1
n=int(input("Enter The Number of Studenst->"))
r=[]
for i in range(n):
    student_roll = int(input("Enter the roll number to search-->"))
    r.append(student_roll)
    sorted_roll_numbers=sorted(r)
print("Sorted Array is-->",sorted_roll_numbers)
student_roll_sorted=int(input("Enter The roll no you want to search--
>"))

binary_result, binary_index = binary_search(sorted_roll_numbers,
student_roll_sorted)
fibonacci_result, fibonacci_index =
fibonacci_search(sorted_roll_numbers, student_roll_sorted)

print("\nBinary Search:")
if binary_result:
    print(f"Student with roll number {student_roll_sorted} attended the
training program at index {binary_index}.")
else:
    print(f"Student with roll number {student_roll_sorted} did not
attend the training program.")

print("\nFibonacci Search:")
if fibonacci_result:
    print(f"Student with roll number {student_roll_sorted} attended the
training program at index {fibonacci_index}.")
else:
    print(f"Student with roll number {student_roll_sorted} did not
attend the training program.")
```

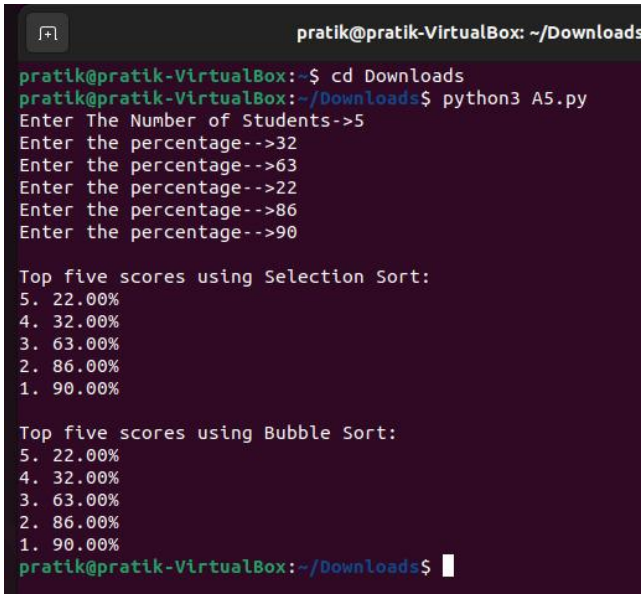




Practical No 5

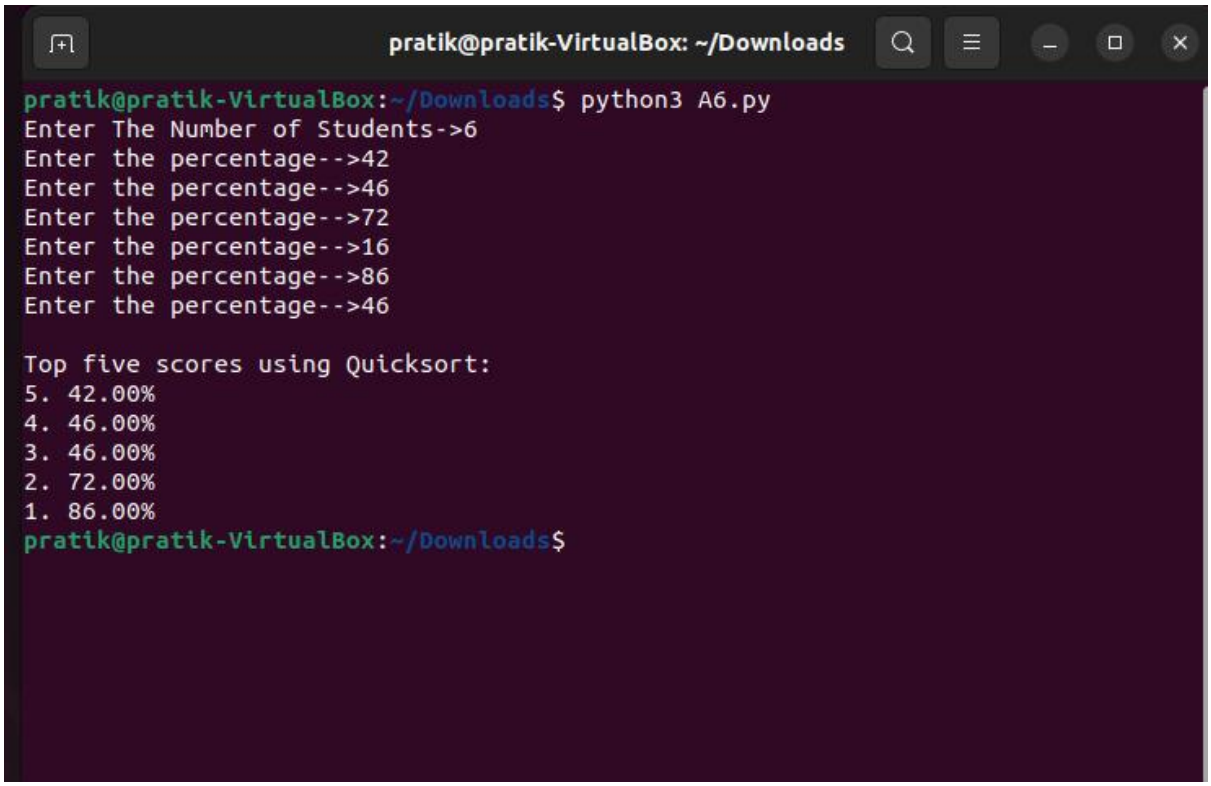
```
def selection_sort(arr):
    n = len(arr)

    for i in range(n - 1):
        min_index = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_index]:
                min_index = j
```

```
        arr[i], arr[min_index] = arr[min_index], arr[i]

def bubble_sort(arr):
    n = len(arr)

    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

n=int(input("Enter The Number of Students->"))
r=[]
for i in range(n):
    percentage = int(input("Enter the percentage-->"))
    r.append(percentage)
selection_sorted_percentages = r.copy()
selection_sort(selection_sorted_percentages)

bubble_sorted_percentages = r.copy()
bubble_sort(bubble_sorted_percentages)

print("\nTop five scores using Selection Sort:")
for i in range(min(5, len(selection_sorted_percentages)), 0, -1):
    print(f"{i}. {selection_sorted_percentages[-i]:.2f}%")

print("\nTop five scores using Bubble Sort:")
for i in range(min(5, len(bubble_sorted_percentages)), 0, -1):
    print(f"{i}. {bubble_sorted_percentages[-i]:.2f}%")
```


pratik@pratik-VirtualBox: ~/Downloads
pratik@pratik-VirtualBox:~$ cd Downloads
pratik@pratik-VirtualBox:~/Downloads$ python3 A5.py
Enter The Number of Students->5
Enter the percentage-->32
Enter the percentage-->63
Enter the percentage-->22
Enter the percentage-->86
Enter the percentage-->90
Top five scores using Selection Sort:
5. 22.00%
4. 32.00%
3. 63.00%
2. 86.00%
1. 90.00%
Top five scores using Bubble Sort:
5. 22.00%
4. 32.00%
3. 63.00%
2. 86.00%
1. 90.00%
pratik@pratik-VirtualBox:~/Downloads$

Practical No 6

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        less_than_pivot = [x for x in arr[1:] if x <= pivot]
        greater_than_pivot = [x for x in arr[1:] if x > pivot]
        return quick_sort(less_than_pivot) + [pivot] +
quick_sort(greater_than_pivot)

n=int(input("Enter The Number of Students->"))
r=[]
for i in range(n):
    percentage= int(input("Enter the percentage-->"))
    r.append(percentage)
sorted_percentages = quick_sort(r)

print("\nTop five scores using Quicksort:")
for i in range(min(5, len(sorted_percentages)), 0, -1):
    print(f"{i}. {sorted_percentages[-i]:.2f}%")
```

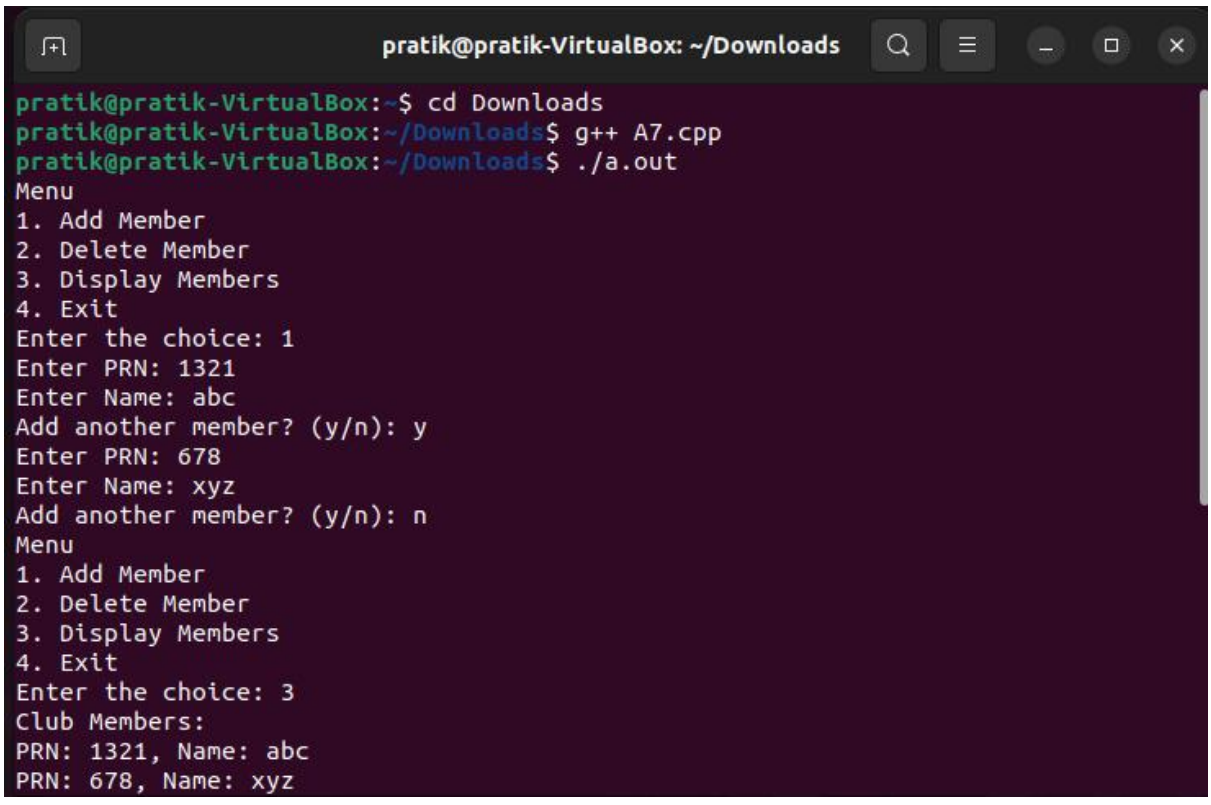


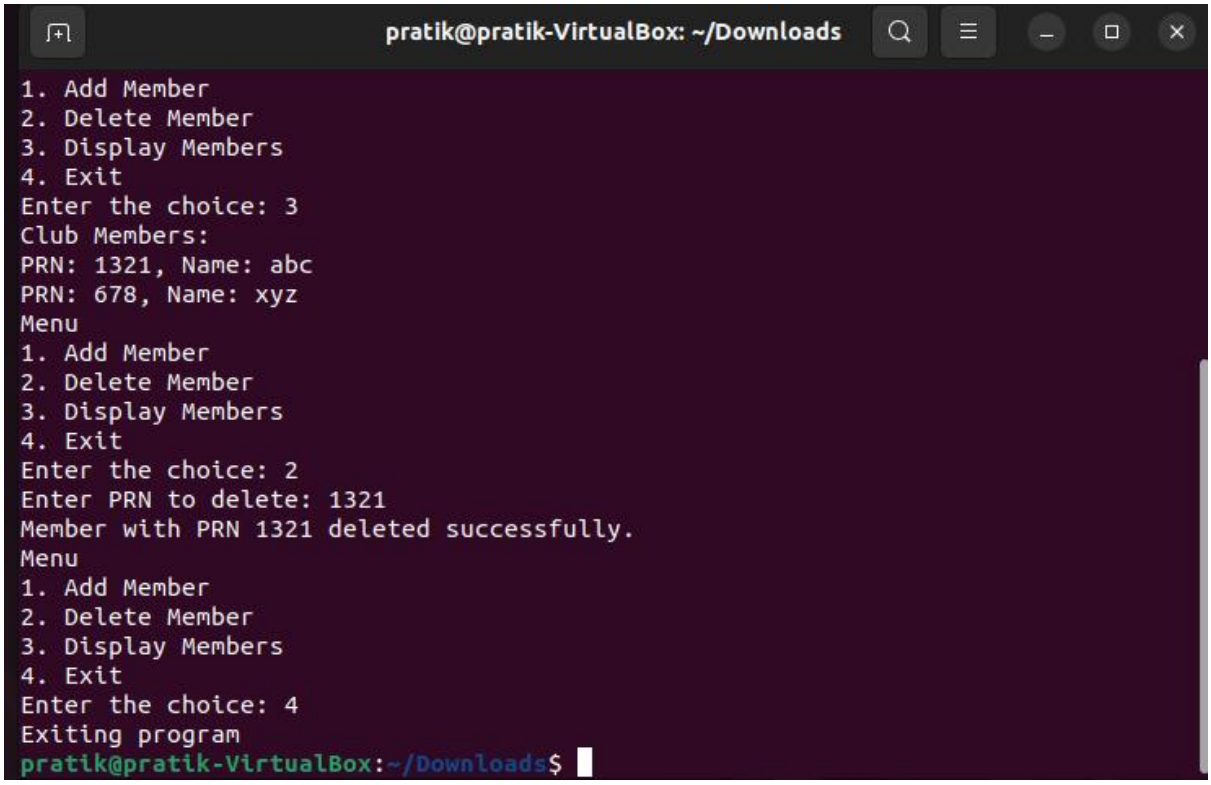
Practical No 7

```
#include <iostream>
#include <string>
using namespace std;

struct ClubMember {
    int prn;
    string name;
    struct ClubMember* link;
};

struct ClubMember* head = NULL;

struct ClubMember* newNode;
struct ClubMember* prevNode;
struct ClubMember* currNode;

int printMenu() {
    int choice;
    cout << "Menu" << endl;
    cout << "1. Add Member" << endl;
    cout << "2. Delete Member" << endl;
    cout << "3. Display Members" << endl;
    cout << "4. Exit" << endl;
    cout << "Enter the choice: ";
    cin >> choice;
    return choice;
}

void addMember() {
    char response;
    do {
        newNode = new struct ClubMember;
        cout << "Enter PRN: ";
        cin >> newNode->prn;
        cout << "Enter Name: ";
        cin>>newNode->name;

        newNode->link = NULL;

        if (head == NULL) {
            head = newNode;
            prevNode = newNode;
        } else {
            prevNode->link = newNode;
            prevNode = newNode;
        }

        cout << "Add another member? (y/n): ";
        cin >> response;
    } while (toupper(response) == 'Y');
}
```

```
void deleteMember() {
    int prnToDelete;
    cout << "Enter PRN to delete: ";
    cin >> prnToDelete;

    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }

    if (head->prn == prnToDelete) {
        currNode = head->link;
        delete head;
        head = currNode;
        cout << "Member with PRN " << prnToDelete << " deleted
successfully." << endl;
        return;
    }

    currNode = head->link;
    prevNode = head;

    while (currNode != NULL) {
        if (currNode->prn == prnToDelete) {
            prevNode->link = currNode->link;
            delete currNode;
            cout << "Member with PRN " << prnToDelete << " deleted
successfully." << endl;
            return;
        } else {
            prevNode = currNode;
            currNode = currNode->link;
        }
    }

    cout << "Member with PRN " << prnToDelete << " not found." << endl;
}

void displayMembers() {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }

    cout << "Club Members:" << endl;
    currNode = head;

    while (currNode != NULL) {
        cout << "PRN: " << currNode->prn << ", Name: " << currNode-
>name << endl;
        currNode = currNode->link;
    }
}

int main() {
    int choice;
    do {
        choice = printMenu();
```

```
        switch (choice) {
            case 1:
                addMember();
                break;
            case 2:
                deleteMember();
                break;
            case 3:
                displayMembers();
                break;
            case 4:
                cout << "Exiting program" << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 4);

    return 0;
}
```




pratik@pratik-VirtualBox: ~/Downloads
1. Add Member
2. Delete Member
3. Display Members
4. Exit
Enter the choice: 3
Club Members:
PRN: 1321, Name: abc
PRN: 678, Name: xyz
Menu
1. Add Member
2. Delete Member
3. Display Members
4. Exit
Enter the choice: 2
Enter PRN to delete: 1321
Member with PRN 1321 deleted successfully.
Menu
1. Add Member
2. Delete Member
3. Display Members
4. Exit
Enter the choice: 4
Exiting program
pratik@pratik-VirtualBox:~/Downloads$

Practical No 8

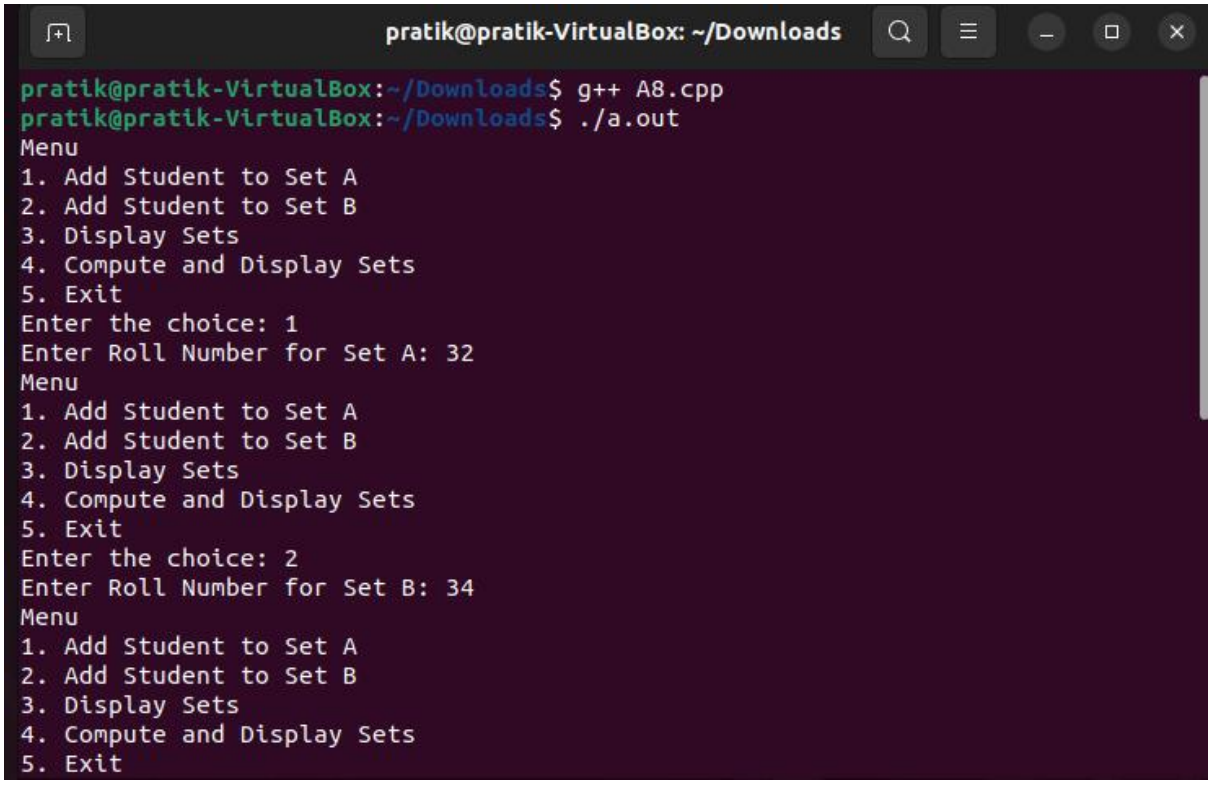
```
#include<iostream>
using namespace std;

struct Student {
   int rollNumber;
   struct Student* link;
};

struct Student* setA = NULL;
struct Student* setB = NULL;

struct Student* newNode;
struct Student* prevNode;
struct Student* currNode;

int printMenu() {
   int choice;
   cout << "Menu" << endl;
   cout << "1. Add Student to Set A" << endl;
   cout << "2. Add Student to Set B" << endl;
   cout << "3. Display Sets" << endl;
   cout << "4. Compute and Display Sets" << endl;
   cout << "5. Exit" << endl;
   cout << "Enter the choice: ";
   cin >> choice;
```

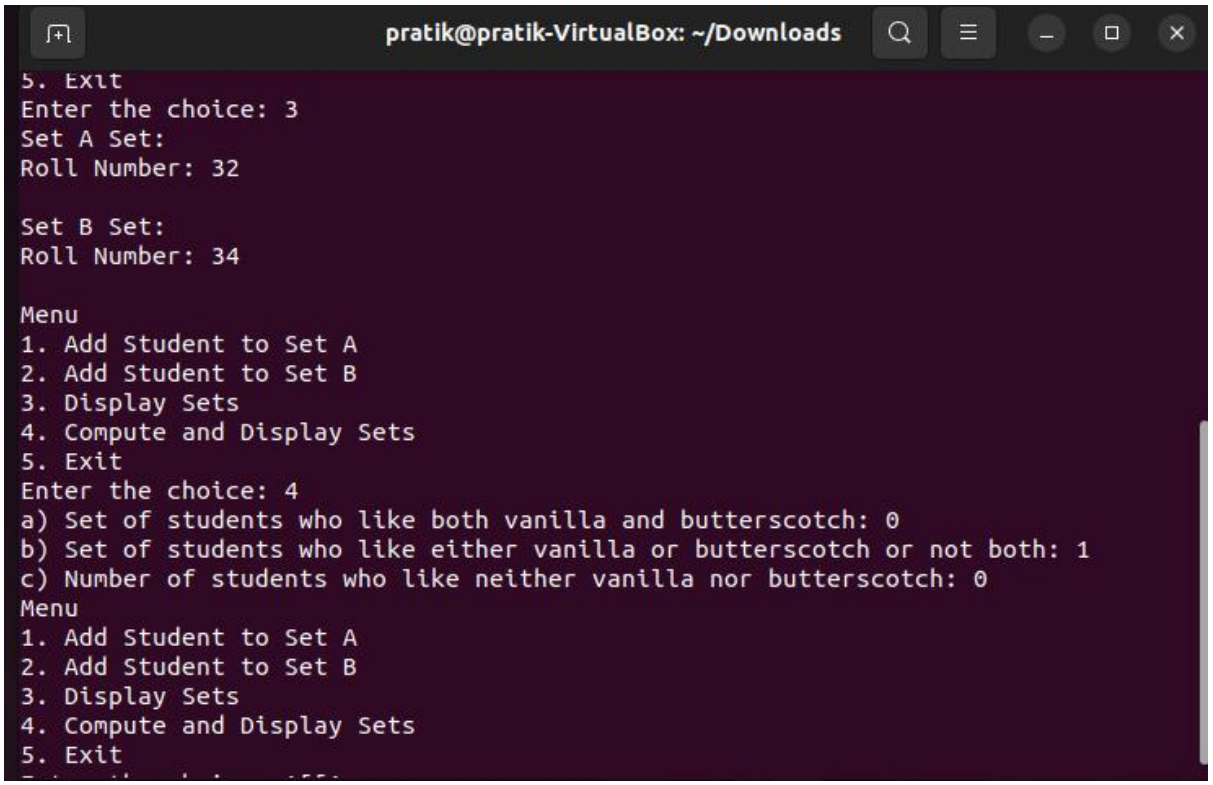
```
    return choice;
}

void addToSet(struct Student*& head, int rollNumber) {
    struct Student* newNode = new struct Student;
    newNode->rollNumber = rollNumber;
    newNode->link = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        struct Student* temp = head;
        while (temp->link != NULL) {
            temp = temp->link;
        }
        temp->link = newNode;
    }
}

void displaySet(struct Student* head, const string& setName) {
    cout << setName << " Set:" << endl;
    while (head != NULL) {
        cout << "Roll Number: " << head->rollNumber << endl;
        head = head->link;
    }
    cout << endl;
}

void computeAndDisplaySets() {
```

```
int bothLikeCount = 0;
int eitherLikeCount = 0;
int neitherLikeCount;

struct Student* tempA = setA;
struct Student* tempB;

while (tempA != NULL) {
   tempB = setB;
   while (tempB != NULL) {
      if (tempA->rollNumber == tempB->rollNumber) {
         bothLikeCount++;
         break;
      }
      tempB = tempB->link;
   }

   eitherLikeCount++;
   tempA = tempA->link;
}


neitherLikeCount = 0;
while (tempA != nullptr) {
   tempB = setB;
   bool likesEither = false;
   while (tempB != nullptr) {
      if (tempA->rollNumber == tempB->rollNumber) {
         likesEither = true;
```

```
            break;
        }
        tempB = tempB->link;
    }

    if (!likesEither) {
        neitherLikeCount++;
    }

    tempA = tempA->link;
  }

  cout << "a) Set of students who like both vanilla and butterscotch: " << bothLikeCount << endl;
  cout << "b) Set of students who like either vanilla or butterscotch or not both: " << eitherLikeCount << endl;
  cout << "c) Number of students who like neither vanilla nor butterscotch: " << neitherLikeCount << endl;
}

int main() {
  int choice;
  do {
    choice = printMenu();
    switch (choice) {
      case 1:
        int rollA;
        cout << "Enter Roll Number for Set A: ";
        cin >> rollA;
```

```
            addToSet(setA, rollA);
            break;
        case 2:
            int rollB;
            cout << "Enter Roll Number for Set B: ";
            cin >> rollB;
            addToSet(setB, rollB);
            break;
        case 3:
            displaySet(setA, "Set A");
            displaySet(setB, "Set B");
            break;
        case 4:
            computeAndDisplaySets();
            break;
        case 5:
            cout << "Exiting program" << endl;
            break;
        default:
            cout << "Invalid choice. Please try again." << endl;
      }
   } while (choice != 5);

   return 0;
}
```

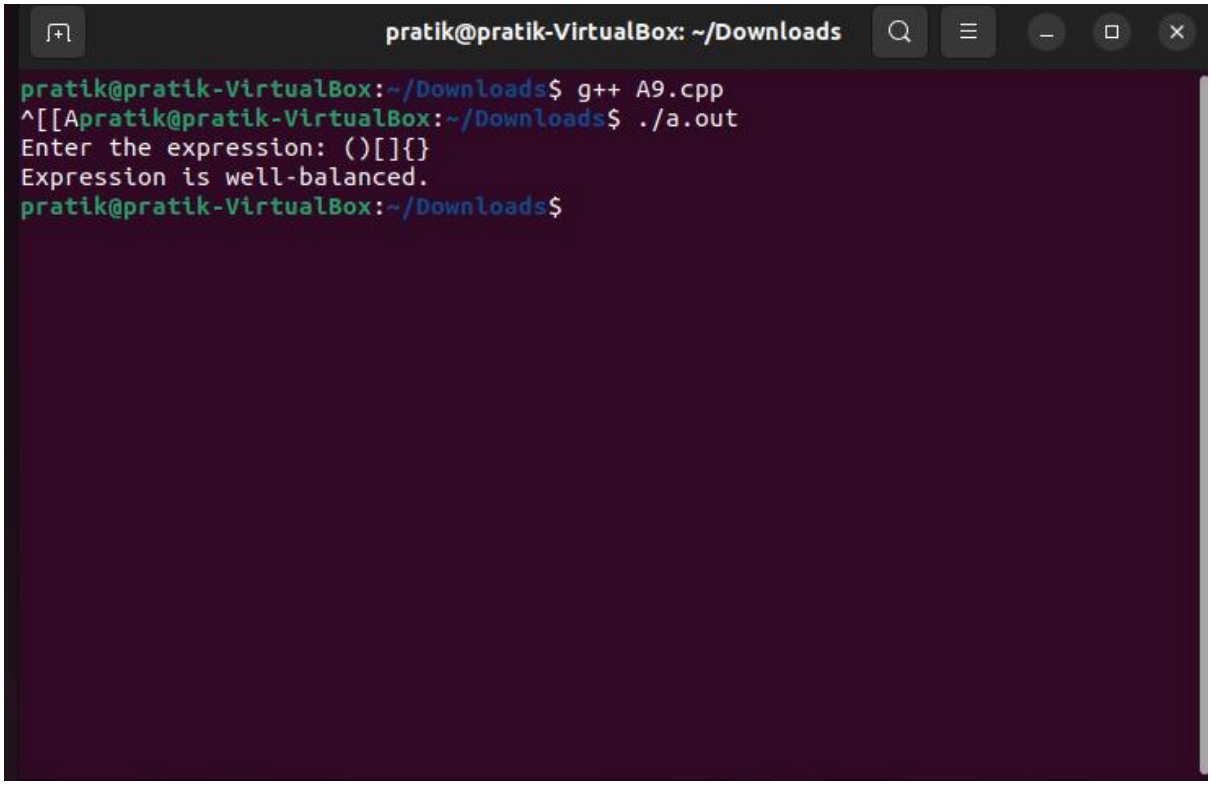

pratik@pratik-VirtualBox: ~/Downloads
pratik@pratik-VirtualBox:~/Downloads$ g++ A8.cpp
pratik@pratik-VirtualBox:~/Downloads$ ./a.out
Menu
1. Add Student to Set A
2. Add Student to Set B
3. Display Sets
4. Compute and Display Sets
5. Exit
Enter the choice: 1
Enter Roll Number for Set A: 32
Menu
1. Add Student to Set A
2. Add Student to Set B
3. Display Sets
4. Compute and Display Sets
5. Exit
Enter the choice: 2
Enter Roll Number for Set B: 34
Menu
1. Add Student to Set A
2. Add Student to Set B
3. Display Sets
4. Compute and Display Sets
5. Exit


pratik@pratik-VirtualBox: ~/Downloads
5. Exit
Enter the choice: 3
Set A Set:
Roll Number: 32
Set B Set:
Roll Number: 34
Menu
1. Add Student to Set A
2. Add Student to Set B
3. Display Sets
4. Compute and Display Sets
5. Exit
Enter the choice: 4
a) Set of students who like both vanilla and butterscotch: 0
b) Set of students who like either vanilla or butterscotch or not both: 1
c) Number of students who like neither vanilla nor butterscotch: 0
Menu
1. Add Student to Set A
2. Add Student to Set B
3. Display Sets
4. Compute and Display Sets
5. Exit

Practical No 9

```
#include<iostream>
#include<cstring>
using namespace std;

#define mx 100

char stk[mx];
int stkempty = 1;
int top = -1;
char expr[mx];

void push(char);
char pop();
bool isMatchingPair(char char1, char char2);

int main() {
    cout << "Enter the expression: ";
    cin.getline(expr, mx);

    for (int i = 0; i < strlen(expr); i++) {
        if (expr[i] == '(' || expr[i] == '{' || expr[i] == '[') {
            push(expr[i]);
        } else if (expr[i] == ')' || expr[i] == '}' || expr[i] == ']')
{
            if (stkempty) {
                cout << "Unmatched " << expr[i] << " at position " << i
+ 1 << endl;
                return 0;
            } else {
                char popped = pop();
                if (!isMatchingPair(popped, expr[i])) {
                    cout << "Mismatched pair at position " << i + 1 <<
": " << popped << " and " << expr[i] << endl;
                    return 0;
                }
            }
        }
    }

    if (stkempty) {
        cout << "Expression is well-balanced." << endl;
    } else {
        cout << "Unmatched " << stk[top] << " in the expression." <<
endl;
    }

    return 0;
}

void push(char x) {
    if (top == mx - 1) {
        cout << "Stack Overflow\n";
```

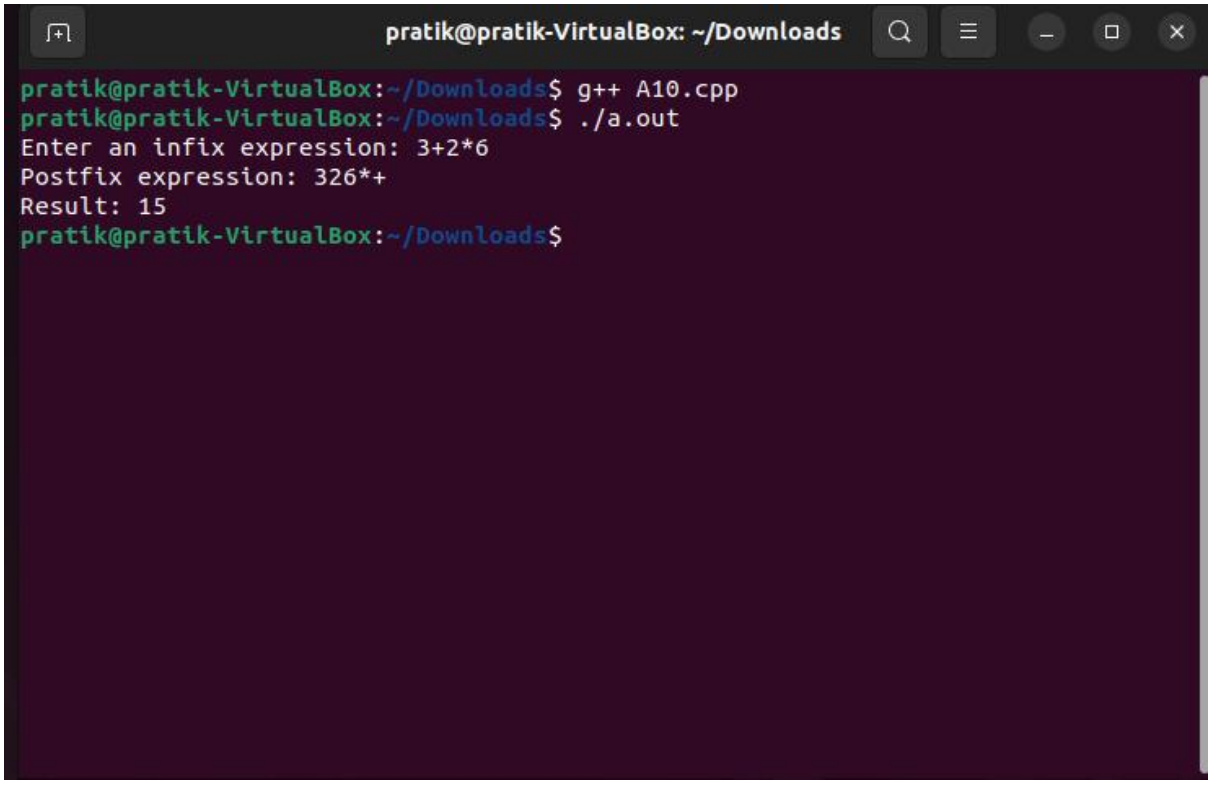
```
        exit(1);
    }
    stk[++top] = x;
    stkempty = 0;  // Update stkempty
}

char pop() {
    if (top == -1) {
        cout << "Stack Underflow\n";
        exit(1);
    }
    char c = stk[top--];
    if (top == -1) {
        stkempty = 1;  // Update stkempty
    }
    return c;
}

bool isMatchingPair(char char1, char char2) {
    return (char1 == '(' && char2 == ')') || (char1 == '{' && char2 ==
'}') || (char1 == '[' && char2 == ']');
}
```



Practical No 10

```
#include <iostream>
#include <stack>
#include <cmath>
using namespace std;

bool isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}

int getPrecedence(char op) {
    if (op == '+' || op == '-')
        return 1;
    if (op == '*' || op == '/')
        return 2;
    return 0;
}

string infixToPostfix(const string& infix) {
    stack<char> s;
    string postfix = "";

    for (char ch : infix) {
        if (isalnum(ch)) {
            postfix += ch;
        } else if (ch == '(') {
            s.push(ch);
        } else if (ch == ')') {
            while (!s.empty() && s.top() != '(') {
                postfix += s.top();
                s.pop();
            }
            if (!s.empty() && s.top() == '(') {
                s.pop();
            }
        } else {
            while (!s.empty() && getPrecedence(ch) <=
getPrecedence(s.top())) {
                postfix += s.top();
                s.pop();
            }
            s.push(ch);
        }
    }

    while (!s.empty()) {
        postfix += s.top();
        s.pop();
    }

    return postfix;
}

double evaluatePostfix(const string& postfix) {
    stack<double> s;

    for (char ch : postfix) {
```

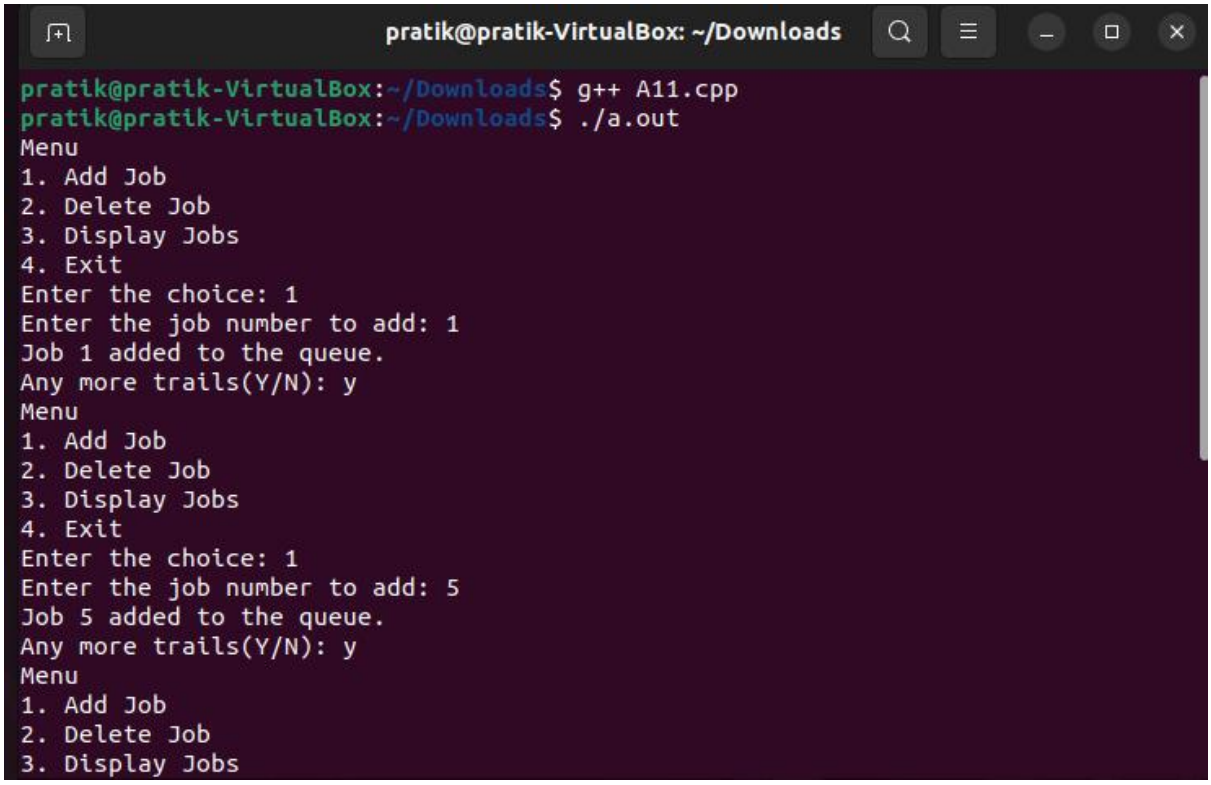
```
        if (isalnum(ch)) {
            s.push(ch - '0');
        } else {
            double operand2 = s.top();
            s.pop();
            double operand1 = s.top();
            s.pop();

            switch (ch) {
                case '+':
                    s.push(operand1 + operand2);
                    break;
                case '-':
                    s.push(operand1 - operand2);
                    break;
                case '*':
                    s.push(operand1 * operand2);
                    break;
                case '/':
                    if (operand2 == 0) {
                        cerr << "Error: Division by zero\n";
                        exit(1);
                    }
                    s.push(operand1 / operand2);
                    break;
            }
        }
    }

    return s.top();
}

int main() {
    string infixExpression;
    cout << "Enter an infix expression: ";
    cin >> infixExpression;

    string postfixExpression = infixToPostfix(infixExpression);
    cout << "Postfix expression: " << postfixExpression << endl;

    double result = evaluatePostfix(postfixExpression);
    cout << "Result: " << result << endl;

    return 0;
}
```

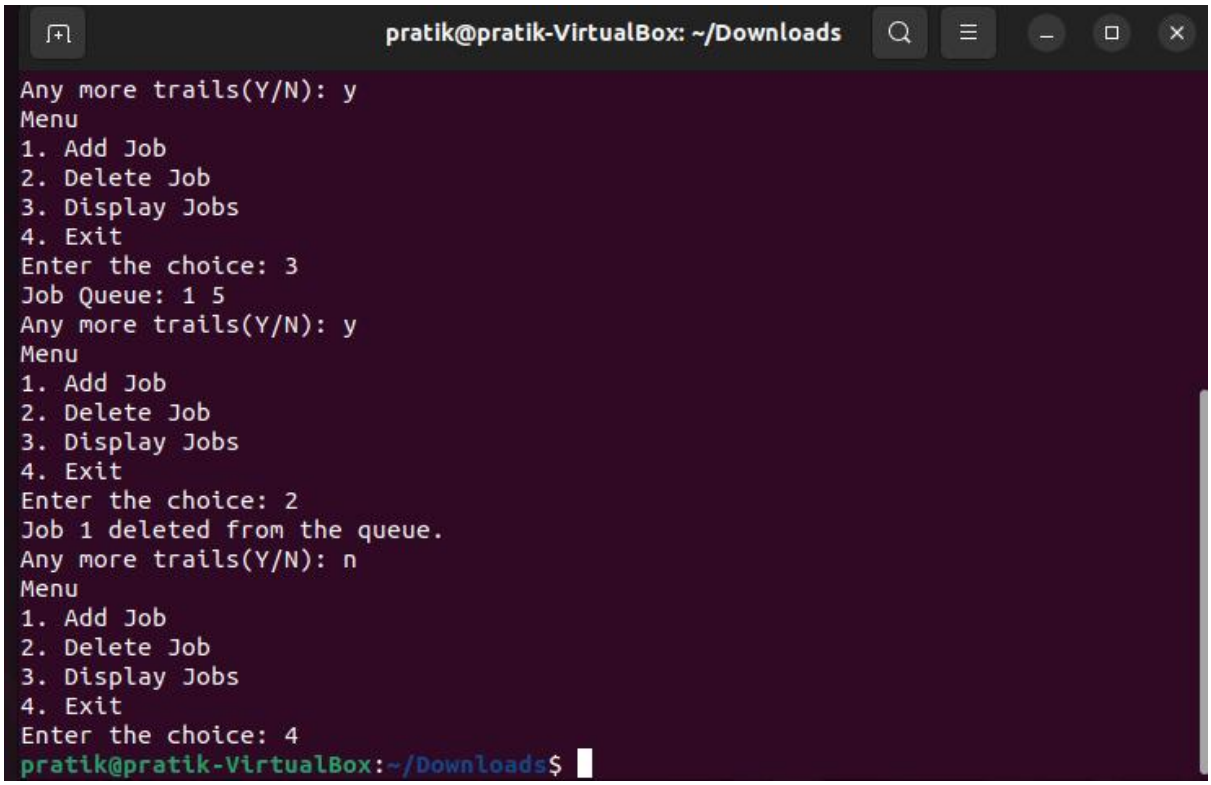

pratik@pratik-VirtualBox: ~/Downloads
pratik@pratik-VirtualBox:~/Downloads$ g++ A10.cpp
pratik@pratik-VirtualBox:~/Downloads$ ./a.out
Enter an infix expression: 3+2*6
Postfix expression: 326*+
Result: 15
pratik@pratik-VirtualBox:~/Downloads$

Practical 11

```
#include <iostream>
using namespace std;

#define mx 3
int q[mx];
int rear = -1, front = -1;

bool isFull() {
    return (rear + 1) % mx == front;
}

bool isEmpty() {
    return front == -1 && rear == -1;
}

void enqueue(int job) {
    if (isFull()) {
        cout << "Job Queue is Full. Cannot add more jobs.\n";
    } else {
        if (isEmpty()) {
            front = rear = 0;
        } else {
            rear = (rear + 1) % mx;
        }
        q[rear] = job;
        cout << "Job " << job << " added to the queue.\n";
    }
}

void dequeue() {
    if (isEmpty()) {
        cout << "Job Queue is Empty. No jobs to delete.\n";
    } else {
        cout << "Job " << q[front] << " deleted from the queue.\n";
        if (front == rear) {
            front = rear = -1;
        } else {
            front = (front + 1) % mx;
        }
    }
}

void display() {
    if (isEmpty()) {
        cout << "Job Queue is Empty.\n";
    } else {
        int i = front;
        cout << "Job Queue: ";
        do {
            cout << q[i] << " ";
            i = (i + 1) % mx;
        } while (i != (rear + 1) % mx);
        cout << endl;
    }
}
```

```
int main() {
    char resp;
    int ch, job=0;
    while(job!=3)
    {
    do {
        cout << "Menu\n";
        cout << "1. Add Job\n";
        cout << "2. Delete Job\n";
        cout << "3. Display Jobs\n";
        cout << "4. Exit\n";
        cout << "Enter the choice: ";
        cin >> ch;

        switch (ch) {
            case 1:
                cout << "Enter the job number to add: ";
                cin >> job;
                enqueue(job);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(1);
            default:
                cout << "Enter Valid Option\n";
                break;
        }

        cout << "Any more trails(Y/N): ";
        cin >> resp;
    } while (toupper(resp) == 'Y');
    }
    return 0;
}
```


pratik@pratik-VirtualBox: ~/Downloads
pratik@pratik-VirtualBox:~/Downloads$ g++ A11.cpp
pratik@pratik-VirtualBox:~/Downloads$ ./a.out
Menu
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter the choice: 1
Enter the job number to add: 1
Job 1 added to the queue.
Any more trails(Y/N): y
Menu
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter the choice: 1
Enter the job number to add: 5
Job 5 added to the queue.
Any more trails(Y/N): y
Menu
1. Add Job
2. Delete Job
3. Display Jobs

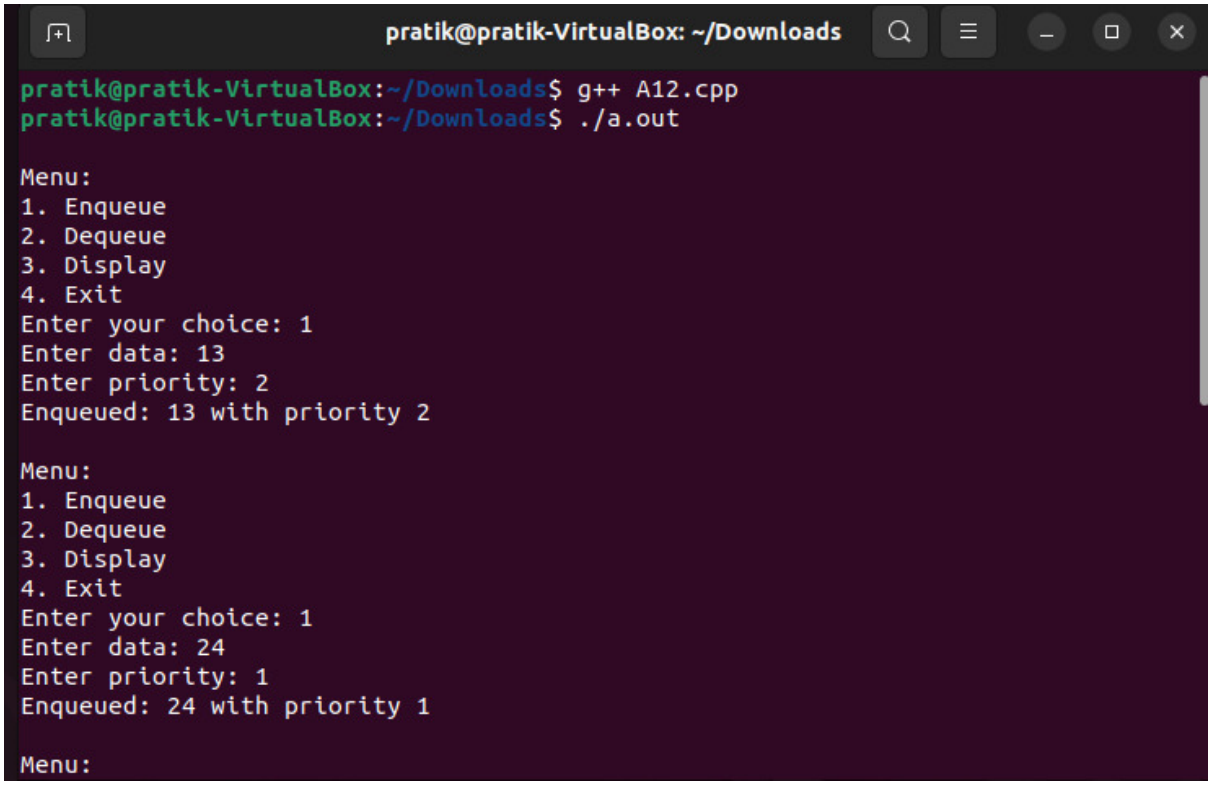

pratik@pratik-VirtualBox: ~/Downloads
Any more trails(Y/N): y
Menu
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter the choice: 3
Job Queue: 1 5
Any more trails(Y/N): y
Menu
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter the choice: 2
Job 1 deleted from the queue.
Any more trails(Y/N): n
Menu
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter the choice: 4
pratik@pratik-VirtualBox:~/Downloads$

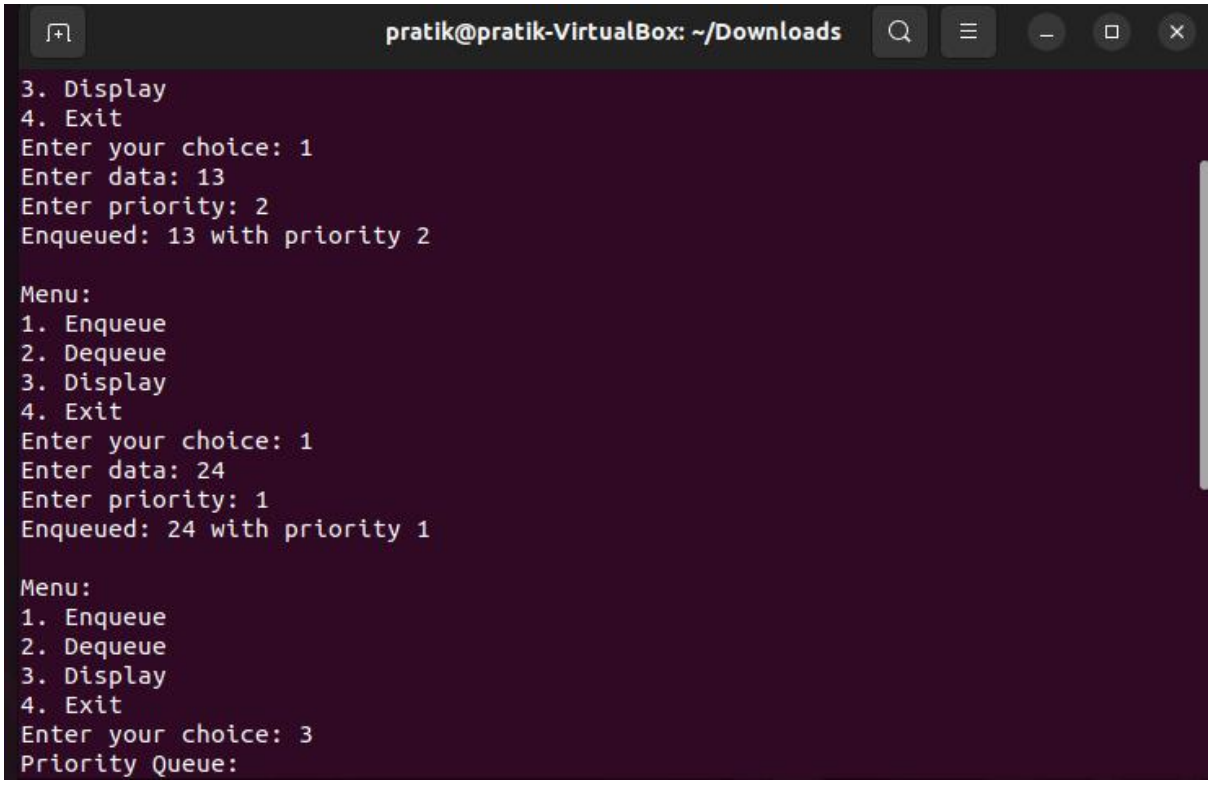
Practical No 12

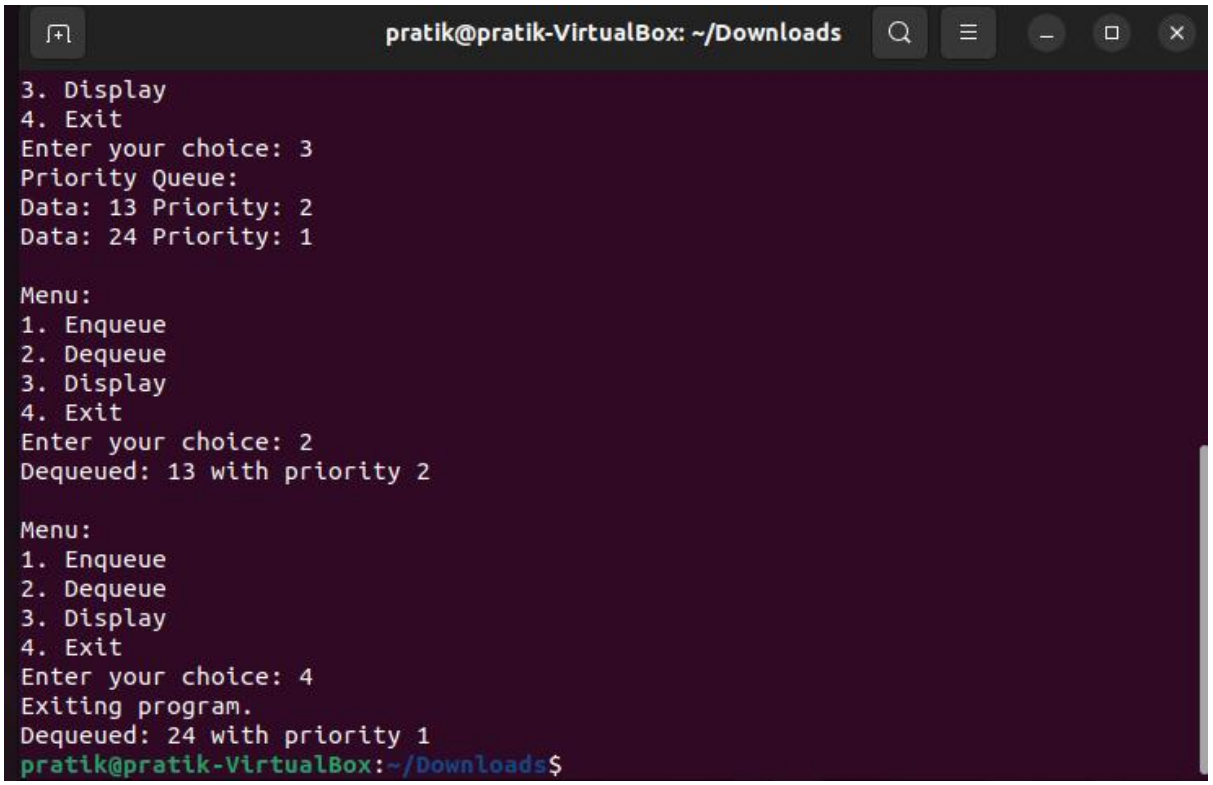
```
#include <iostream>
using namespace std;

// Template class for PriorityQueue
template <typename T>
class PriorityQueue {
private:
    // Inner class to represent each item in the priority queue
    class QueueNode {
    public:
        T data;
        int priority;
        QueueNode* next;

        // Constructor to initialize data and priority
        QueueNode(T value, int prio) : data(value), priority(prio),
next(nullptr) {}
    };

    QueueNode* front; // Front of the priority queue

public:
    // Constructor to initialize the front pointer
    PriorityQueue() : front(nullptr) {}

    // Destructor to deallocate memory
    ~PriorityQueue() {
        while (!isEmpty()) {
            dequeue();
        }
    }

    // Function to check if the priority queue is empty
    bool isEmpty() const {
        return front == nullptr;
    }

    // Function to enqueue an item with a given priority
    void enqueue(T value, int priority) {
        QueueNode* newNode = new QueueNode(value, priority);

        if (isEmpty() || priority > front->priority) {
            newNode->next = front;
            front = newNode;
        } else {
            QueueNode* current = front;
            while (current->next != nullptr && priority <= current-
>next->priority) {
                current = current->next;
            }
            newNode->next = current->next;
            current->next = newNode;
        }

        cout << "Enqueued: " << value << " with priority " << priority
<< endl;
```

```
    }

    // Function to dequeue the highest priority item
    void dequeue() {
        if (isEmpty()) {
            cout << "Priority Queue is empty. Cannot dequeue.\n";
        } else {
            QueueNode* temp = front;
            front = front->next;
            cout << "Dequeued: " << temp->data << " with priority " <<
temp->priority << endl;
            delete temp;
        }
    }

    // Function to display the items in the priority queue
    void display() const {
        if (isEmpty()) {
            cout << "Priority Queue is empty.\n";
        } else {
            cout << "Priority Queue:\n";
            QueueNode* current = front;
            while (current != nullptr) {
                cout << "Data: " << current->data << " Priority: " <<
current->priority << endl;
                current = current->next;
            }
        }
    }
};

int main() {
    PriorityQueue<string> pq; // Example for strings, you can change
the type

    char choice;
    do {
        cout << "\nMenu:\n";
        cout << "1. Enqueue\n";
        cout << "2. Dequeue\n";
        cout << "3. Display\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case '1': {
                string data;
                int priority;

                cout << "Enter data: ";
                cin >> data;

                cout << "Enter priority: ";
                cin >> priority;

                pq.enqueue(data, priority);
                break;
```

```
            }
            case '2':
                pq.dequeue();
                break;
            case '3':
                pq.display();
                break;
            case '4':
                cout << "Exiting program.\n";
                break;
            default:
                cout << "Invalid choice. Try again.\n";
        }

    } while (choice != '4');

    return 0;
}
```




pratik@pratik-VirtualBox: ~/Downloads
3. Display
4. Exit
Enter your choice: 1
Enter data: 13
Enter priority: 2
Enqueued: 13 with priority 2
Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter data: 24
Enter priority: 1
Enqueued: 24 with priority 1
Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Priority Queue:

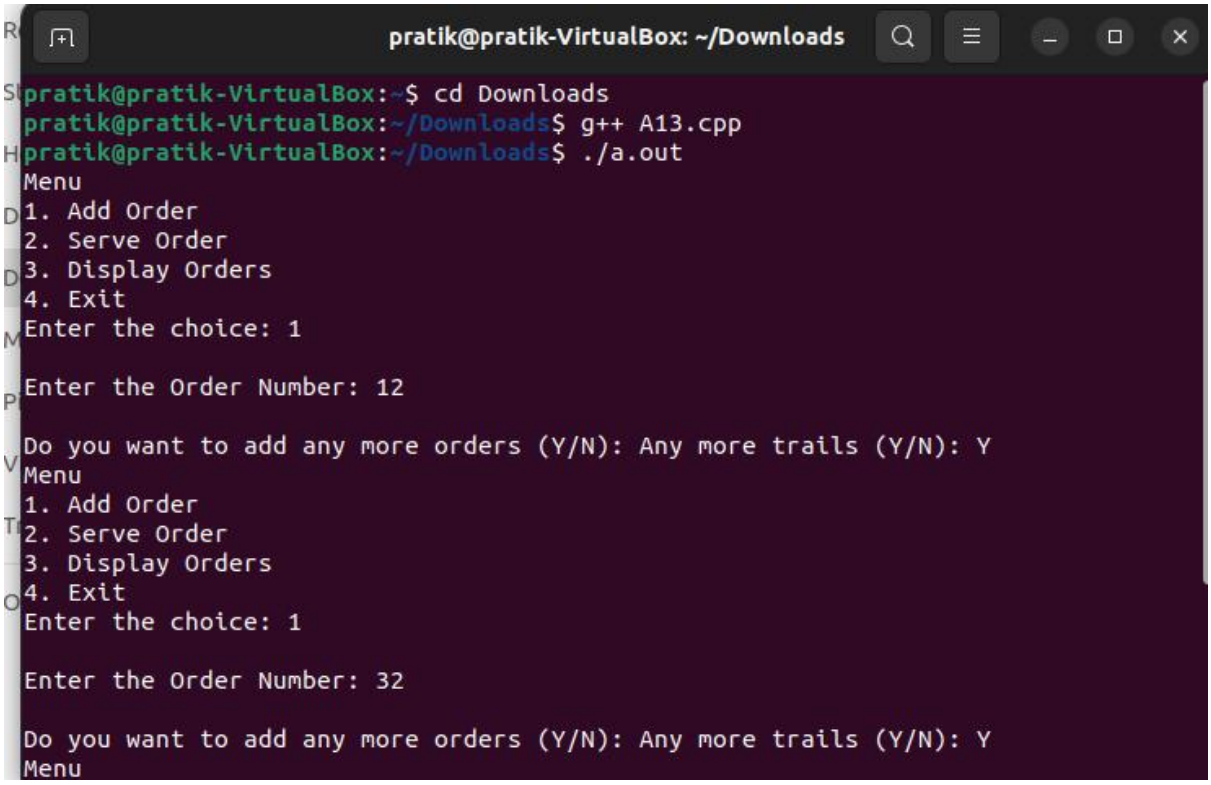

pratik@pratik-VirtualBox: ~/Downloads
3. Display
4. Exit
Enter your choice: 3
Priority Queue:
Data: 13 Priority: 2
Data: 24 Priority: 1
Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued: 13 with priority 2
Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
Exiting program.
Dequeued: 24 with priority 1
pratik@pratik-VirtualBox:~/Downloads$

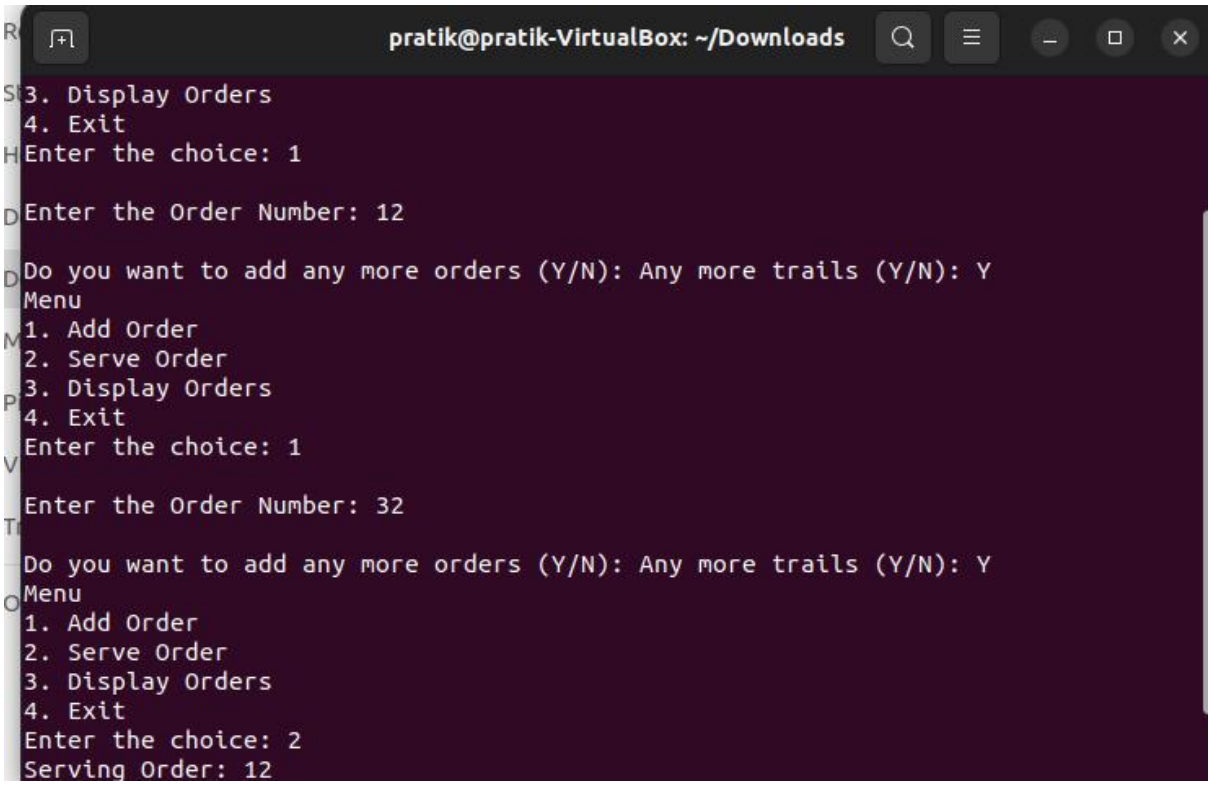
Practical No 13

```
#include<iostream>
#include<conio.h>
#include<ctype.h>
#include<stdlib.h>
using namespace std;

#define mx 3

int q[mx];
int r = -1, f = -1;

int qfull(void);
int qempty(void);
void addq(void);
void delq(void);
void display(void);

int main()
{
    char resp;
    int ch;
    do
    {
        cout << "Menu\n";
        cout << "1. Add Order\n";
        cout << "2. Serve Order\n";
        cout << "3. Display Orders\n";
        cout << "4. Exit\n";
        cout << "Enter the choice: ";
        cin >> ch;

        switch (ch)
        {
        case 1:
            addq();
            break;
        case 2:
            delq();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(1);
        default:
            cout << "Enter Valid Option";
            break;
        }
        cout << "Any more trails (Y/N): ";
        resp = getchar();
    } while (toupper(resp) == 'Y');

    return 0;
}

int qfull(void)
```

```
{
    if (f == (r + 1) % mx)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

int qempty(void)
{
    if ((r == -1 && f == -1))
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

void addq(void)
{
    char resp;
    int orderNumber;

    do
    {
        if (qfull() == 1)
        {
            cout << "Order Queue is Full";
            getch();
        }
        else
        {
            cout << "\nEnter the Order Number: ";
            cin >> orderNumber;
            if (f == -1 && r == -1)
            {
                f = r = 0;
            }
            else
            {
                r = (r + 1) % mx;
            }
            q[r] = orderNumber;
        }
        cout << "\nDo you want to add any more orders (Y/N): ";
        resp = getchar();
    } while (toupper(resp) == 'Y');
}

void delq(void)
{
    char resp;
    do
```

```
    {
        if (qempty() == 1)
        {
            cout << "Order Queue is empty";
            getch();
        }
        else
        {
            if (f == r)
            {
                cout << "Serving Order: " << q[f] << endl;
                f = r = -1;
            }
            else
            {
                cout << "Serving Order: " << q[f] << endl;
                f = (f + 1) % mx;
            }
        }
        getch();
        cout << "Any more Order Servings (Y/N): ";
        resp = getchar();
    } while (toupper(resp) == 'Y');
}

void display(void)
{
    int i;
    i = f;
    if (qempty() == 1)
    {
        cout << "Order Queue is empty";
    }
    else
    {
        cout << "Order Queue is \n";
        while (i != r)
        {
            cout << "\tOrder Number: " << q[i] << endl;
            i = (i + 1) % mx;
        }
        cout << "\tOrder Number: " << q[i] << endl;
        getch();
    }
}
```


pratik@pratik-VirtualBox: ~/Downloads
pratik@pratik-VirtualBox:~$ cd Downloads
pratik@pratik-VirtualBox:~/Downloads$ g++ A13.cpp
pratik@pratik-VirtualBox:~/Downloads$ ./a.out
Menu
1. Add Order
2. Serve Order
3. Display Orders
4. Exit
Enter the choice: 1
Enter the Order Number: 12
Do you want to add any more orders (Y/N): Any more trails (Y/N): Y
Menu
1. Add Order
2. Serve Order
3. Display Orders
4. Exit
Enter the choice: 1
Enter the Order Number: 32
Do you want to add any more orders (Y/N): Any more trails (Y/N): Y
Menu


pratik@pratik-VirtualBox: ~/Downloads
3. Display Orders
4. Exit
Enter the choice: 1
Enter the Order Number: 12
Do you want to add any more orders (Y/N): Any more trails (Y/N): Y
Menu
1. Add Order
2. Serve Order
3. Display Orders
4. Exit
Enter the choice: 1
Enter the Order Number: 32
Do you want to add any more orders (Y/N): Any more trails (Y/N): Y
Menu
1. Add Order
2. Serve Order
3. Display Orders
4. Exit
Enter the choice: 2
Serving Order: 12