Practical no. 1

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

**In second year computer engineering class, group A students play cricket,
group B students play badminton and group C students play football.
Write a python program using functions to compute following: -
a) List of students who play both cricket and badminton
b) List of students who play either cricket or badminton but not both
c) Number of students who play neither cricket nor badminton
d) Number of students who play cricket and football but not badminton.
(Note- While realizing the group, duplicate entries should be avoided, Do not
use SET built-in functions)**

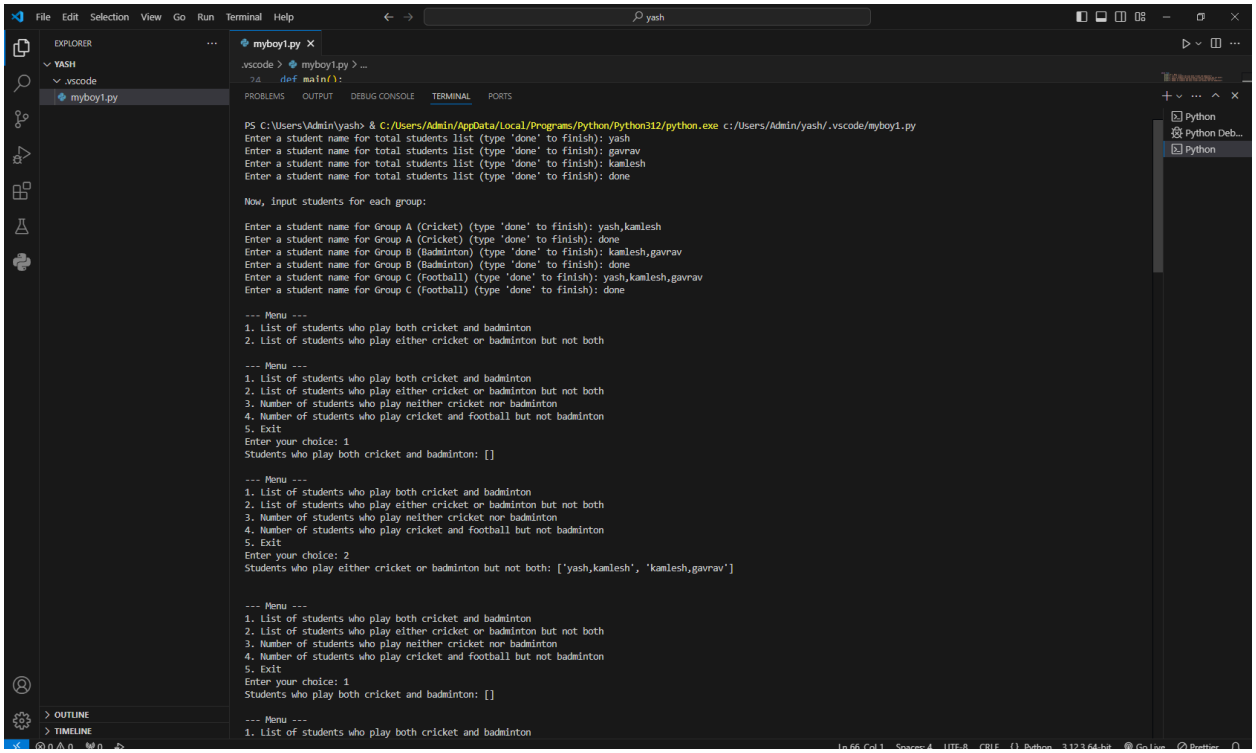
CODE:

```
# Define functions for each task
def input_students(group_name):
    students = []
    while True:
        student = input(f"Enter a student name for {group_name} (type 'done' to finish):
")
        if student.lower() == 'done':
            break
        if student not in students:  # Avoid duplicate entries
            students.append(student)
    return students

def both_cricket_and_badminton(group_a, group_b):
    return [student for student in group_a if student in group_b]

def either_cricket_or_badminton(group_a, group_b):
    return [student for student in group_a + group_b if (student in group_a and
student not in group_b) or (student in group_b and student not in group_a)]
```

```
def neither_cricket_nor_badminton(group_a, group_b, total_students):
    return [student for student in total_students if student not in group_a and student not in group_b]

def cricket_and_football_not_badminton(group_a, group_c, group_b):
    return [student for student in group_a if student in group_c and student not in group_b]

def main():
    total_students = input_students("total students list")
    print("\nNow, input students for each group:\n")
    group_a = input_students("Group A (Cricket)")
    group_b = input_students("Group B (Badminton)")
    group_c = input_students("Group C (Football)")

    while True:
        print("\n--- Menu ---")
        print("1. List of students who play both cricket and badminton")
        print("2. List of students who play either cricket or badminton but not both")
        print("3. Number of students who play neither cricket nor badminton")
        print("4. Number of students who play cricket and football but not badminton")
        print("5. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            result = both_cricket_and_badminton(group_a, group_b)
            print("Students who play both cricket and badminton:", result)

        elif choice == 2:
            result = either_cricket_or_badminton(group_a, group_b)
            print("Students who play either cricket or badminton but not both:", result)

        elif choice == 3:
            result = neither_cricket_nor_badminton(group_a, group_b, total_students)
            print("Number of students who play neither cricket nor badminton:", len(result))

        elif choice == 4:
            result = cricket_and_football_not_badminton(group_a, group_c, group_b)
            print("Number of students who play cricket and football but not badminton:", len(result))

        elif choice == 5:
            print("Exiting program.")
            break
```

```
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:



Practical no. 2

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

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

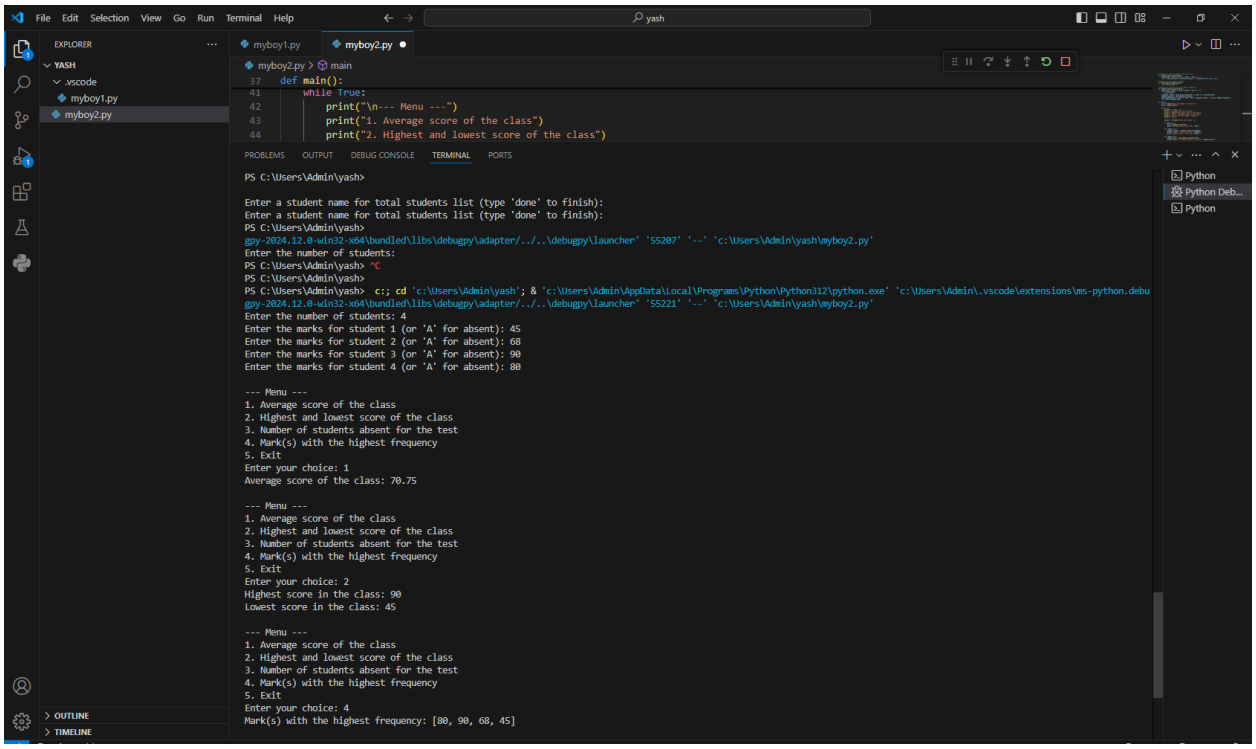
CODE:

```
# Function to input the marks
def input_marks(n):
    marks = []
    for i in range(n):
        mark = input(f"Enter the marks
for student {i + 1} (or 'A' for absent): ")
        if mark.upper() == 'A':
            marks.append('A')
        else:
            marks.append(int(mark))
    return marks

# Function to compute the average
score
def average_score(marks):
    total = sum([mark for mark in marks
if mark != 'A'])
    count = len([mark for mark in marks
if mark != 'A'])
    return total / count if count > 0 else
```

```
# Function to find the highest and
lowest score
def highest_lowest_score(marks):
    valid_marks = [mark for mark in
marks if mark != 'A']
    return (max(valid_marks),
min(valid_marks)) if valid_marks else
(None, None)

# Function to count absent students
def count_absent_students(marks):
    return marks.count('A')

# Function to find the mark with
highest frequency
def highest_frequency_mark(marks):
    valid_marks = [mark for mark in
marks if mark != 'A']
    if not valid_marks:
        return None
    frequency = {mark:
valid_marks.count(mark) for mark in
set(valid_marks)}
    highest_frequency =
max(frequency.values())
    most_frequent_marks = [mark for
mark, freq in frequency.items() if freq
== highest_frequency]
    return most_frequent_marks

def main():
    n = int(input("Enter the number of
students: "))
    marks = input_marks(n)

    while True:
        print("\n--- Menu ---")
        print("1. Average score of the
class")
        print("2. Highest and lowest score
of the class")
        print("3. Number of students
absent for the test")
```

```
        print("4. Mark(s) with the highest
frequency")
        print("5. Exit")

        choice = int(input("Enter your
choice: "))

        if choice == 1:
            avg = average_score(marks)
            print(f"Average score of the
class: {avg}")

        elif choice == 2:
            highest, lowest =
highest_lowest_score(marks)
            print(f"Highest score in the
class: {highest}")
            print(f"Lowest score in the
class: {lowest}")

        elif choice == 3:
            absent_count =
count_absent_students(marks)
            print(f"Number of students
absent for the test: {absent_count}")

        elif choice == 4:
            most_frequent_marks =
highest_frequency_mark(marks)
            print(f"Mark(s) with the highest
frequency: {most_frequent_marks}")

        elif choice == 5:
            print("Exiting program.")
            break

        else:
            print("Invalid choice. Please try
again.")

if __name__ == "__main__":
    main()
```

OUTPUT:


File Edit Selection View Go Run Terminal Help
yash
EXPLORER
YASH
.vscode
myboy1.py
myboy2.py
myboy1.py
myboy2.py
myboy2.py > main
37 def main():
41 while True:
42 print("\n--- Menu ---")
43 print("1. Average score of the class")
44 print("2. Highest and lowest score of the class")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Admin\yash>
Enter a student name for total students list (type 'done' to finish):
Enter a student name for total students list (type 'done' to finish):
PS C:\Users\Admin\yash>
gpy-2024.12.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '55207' '--' 'c:\Users\Admin\yash\myboy2.py'
Enter the number of students:
PS C:\Users\Admin\yash> ^C
PS C:\Users\Admin\yash>
PS C:\Users\Admin\yash> c:; cd 'c:\Users\Admin\yash'; & 'c:\Users\Admin\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debu
gpy-2024.12.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '55221' '--' 'c:\Users\Admin\yash\myboy2.py'
Enter the number of students: 4
Enter the marks for student 1 (or 'A' for absent): 45
Enter the marks for student 2 (or 'A' for absent): 68
Enter the marks for student 3 (or 'A' for absent): 90
Enter the marks for student 4 (or 'A' for absent): 80
--- Menu ---
1. Average score of the class
2. Highest and lowest score of the class
3. Number of students absent for the test
4. Mark(s) with the highest frequency
5. Exit
Enter your choice: 1
Average score of the class: 70.75
--- Menu ---
1. Average score of the class
2. Highest and lowest score of the class
3. Number of students absent for the test
4. Mark(s) with the highest frequency
5. Exit
Enter your choice: 2
Highest score in the class: 90
Lowest score in the class: 45
--- Menu ---
1. Average score of the class
2. Highest and lowest score of the class
3. Number of students absent for the test
4. Mark(s) with the highest frequency
5. Exit
Enter your choice: 4
Mark(s) with the highest frequency: [80, 90, 68, 45]
Python
Python Deb...
Python
OUTLINE
TIMELINE

Practical no. 3

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

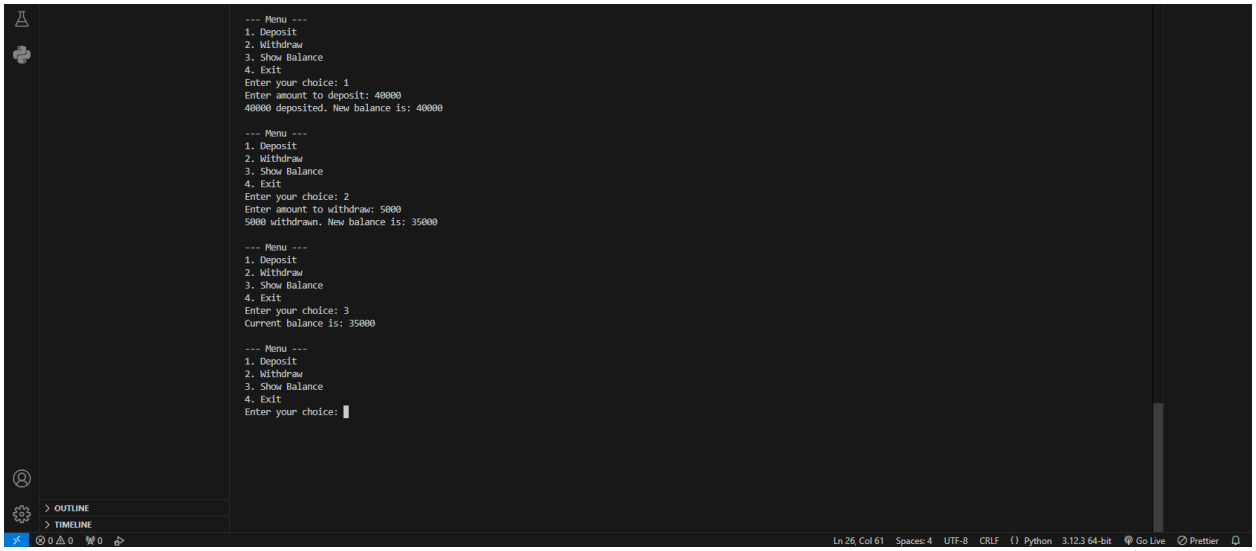
Write a Python program that computes the net amount of a bank account based
a transaction log from console input. The transaction log format is shown as
following: D 100 W 200 (Withdrawal is not allowed if balance is going negative. Write functions for withdraw and deposit) D means deposit while W
means withdrawal.
Suppose the following input is supplied to the program:
D 300, D 300 , W 200, D 100 Then, the output should be: 500

CODE:

```
# Function for deposit
def deposit(balance, amount):
    return balance + amount

# Function for withdrawal
def withdraw(balance, amount):
    if balance >= amount:
        return balance - amount
    else:
        print("Insufficient funds for withdrawal.")
        return balance

def main():
    balance = 0

    while True:
```

```
        print("\n--- Menu ---")
        print("1. Deposit")
        print("2. Withdraw")
        print("3. Show Balance")
        print("4. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            amount = int(input("Enter amount to deposit: "))
            balance = deposit(balance, amount)
            print(f"{amount} deposited. New balance is: {balance}")

        elif choice == 2:
            amount = int(input("Enter amount to withdraw: "))
            balance = withdraw(balance, amount)
            print(f"{amount} withdrawn. New balance is: {balance}")

        elif choice == 3:
            print(f"Current balance is: {balance}")

        elif choice == 4:
            print("Exiting program.")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:



Practical no. 4

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

Write a python program to store second year percentage of students in array.
Write function for sorting array of floating point numbers in ascending order
using
a) Insertion sort
b) Shell Sort and display top five scores

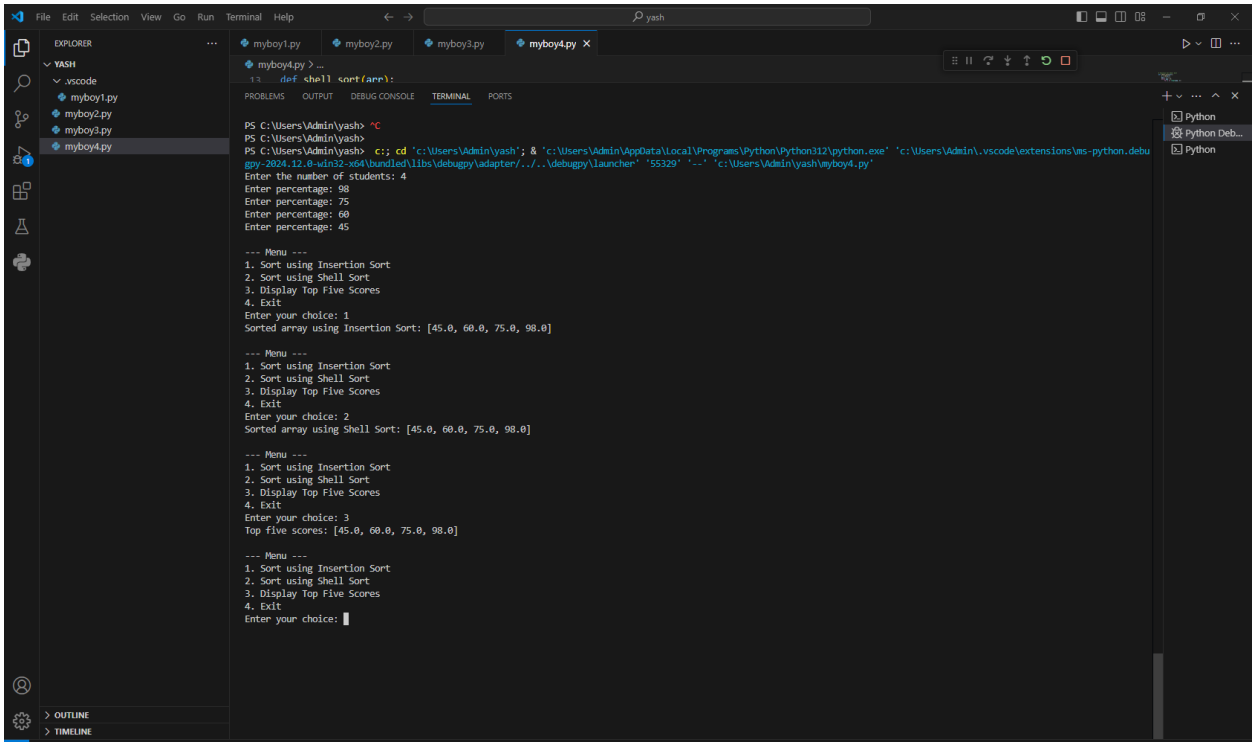
CODE:

```
# Function for Insertion Sort
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
```

```
    return arr

# Function for Shell Sort
def shell_sort(arr):
    n = len(arr)
    gap = n // 2
    while gap > 0:
        for i in range(gap, n):
            temp = arr[i]
            j = i
            while j >= gap and arr[j - gap] > temp:
                arr[j] = arr[j - gap]
                j -= gap
            arr[j] = temp


   gap //= 2
    return arr

# Function to get top five scores
def top_five_scores(arr):
    return arr[-5:]

def main():
    # Store second year percentage of students in an array
    num_students = int(input("Enter the number of students: "))
    percentages = []
    for _ in range(num_students):
        percentage = float(input("Enter percentage: "))
        percentages.append(percentage)

    while True:
        print("\n--- Menu ---")
        print("1. Sort using Insertion Sort")
        print("2. Sort using Shell Sort")
        print("3. Display Top Five Scores")
        print("4. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            sorted_percentages = insertion_sort(percentages.copy())
            print("Sorted array using Insertion Sort:", sorted_percentages)

        elif choice == 2:
            sorted_percentages = shell_sort(percentages.copy())
```

```
            print("Sorted array using Shell Sort:", sorted_percentages)

        elif choice == 3:
            sorted_percentages = sorted(percentages.copy())
            top_scores = top_five_scores(sorted_percentages)
            print("Top five scores:", top_scores)

        elif choice == 4:
            print("Exiting program.")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:



Practical no. 5

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

**Write a python program to store first year percentage of students in array. Write
function for sorting array of floating point numbers in ascending order using
a) Selection Sort
b) Bubble sort and display top five scores**

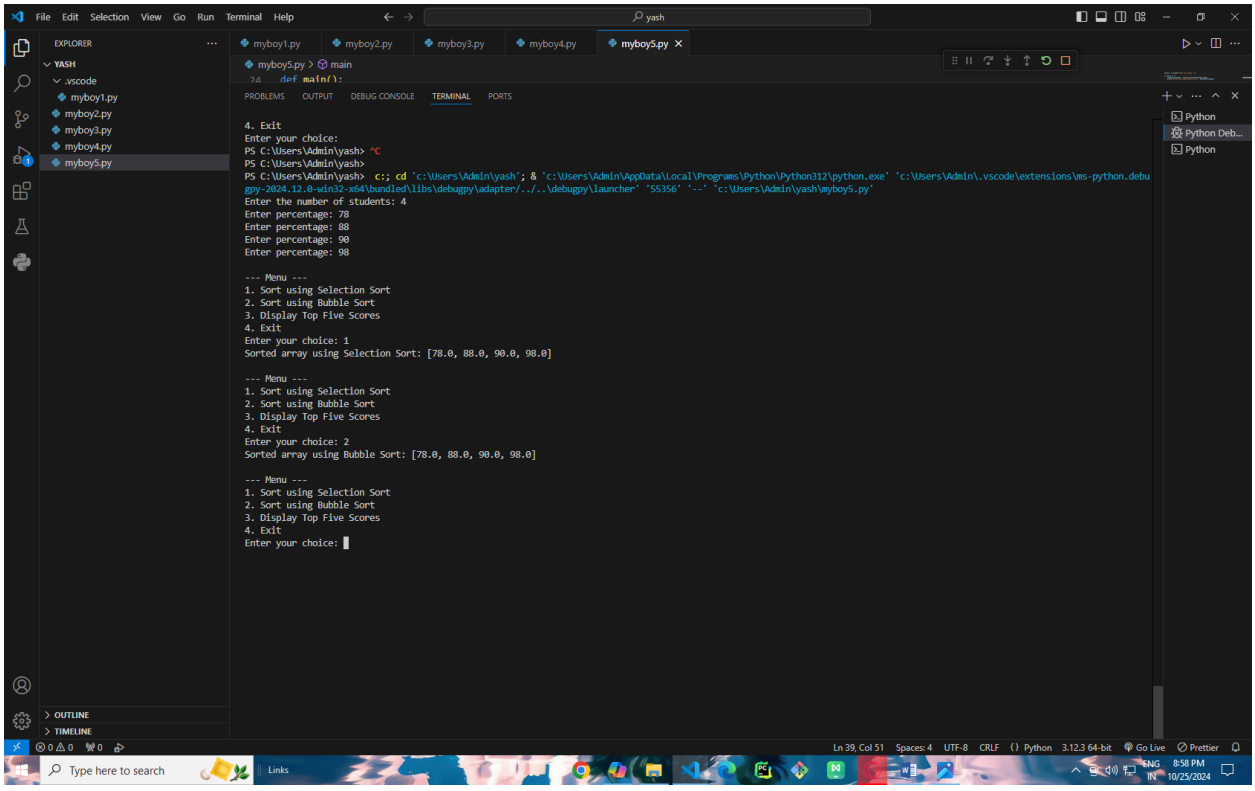
CODE:

```
# Function for Selection Sort
def selection_sort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i + 1, len(arr)):
            if arr[min_idx] > arr[j]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

```
# Function for Bubble Sort
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# Function to get top five scores
def top_five_scores(arr):
    return arr[-5:]



def main():
    # Store first year percentage of students in an array
    num_students = int(input("Enter the number of students: "))
    percentages = []
    for _ in range(num_students):
        percentage = float(input("Enter percentage: "))
        percentages.append(percentage)

    while True:
        print("\n--- Menu ---")
        print("1. Sort using Selection Sort")
        print("2. Sort using Bubble Sort")
        print("3. Display Top Five Scores")
        print("4. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            sorted_percentages = selection_sort(percentages.copy())
            print("Sorted array using Selection Sort:", sorted_percentages)

        elif choice == 2:
            sorted_percentages = bubble_sort(percentages.copy())
            print("Sorted array using Bubble Sort:", sorted_percentages)

        elif choice == 3:
            sorted_percentages = sorted(percentages.copy())
            top_scores = top_five_scores(sorted_percentages)
            print("Top five scores:", top_scores)
```

```
        elif choice == 4:
            print("Exiting program.")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:



Practical no. 6

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

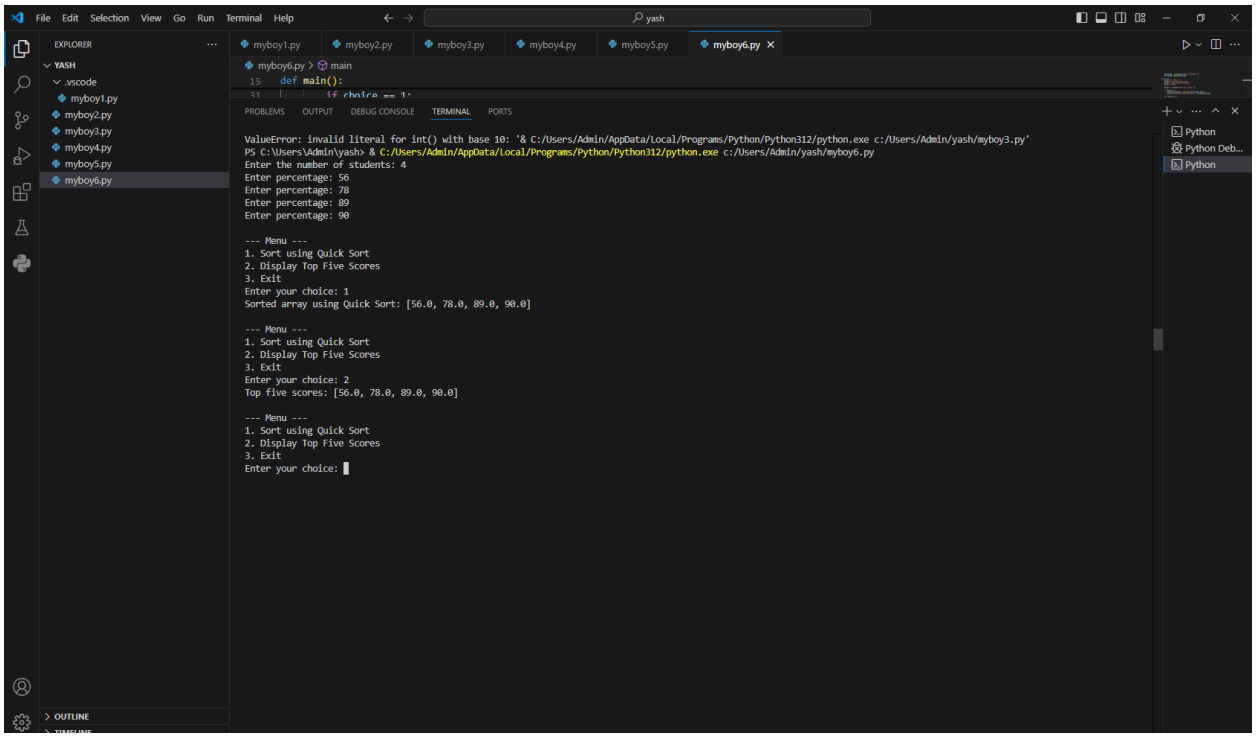
**Write a python program to store first year percentage of students in array. Write
function for sorting array of floating point numbers in ascending order using
quick sort and display top five scores.**

CODE:

```
# Function for Quick Sort
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

# Function to get top five scores
def top_five_scores(arr):
```

```
    return arr[-5:]

def main():
    # Store first year percentage of students in an array
    num_students = int(input("Enter the number of students: "))
    percentages = []
    for _ in range(num_students):
        percentage = float(input("Enter percentage: "))
        percentages.append(percentage)

    while True:
        print("\n--- Menu ---")
        print("1. Sort using Quick Sort")




    print("2. Display Top Five Scores")
        print("3. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            sorted_percentages = quick_sort(percentages.copy())
            print("Sorted array using Quick Sort:", sorted_percentages)

        elif choice == 2:
            sorted_percentages = quick_sort(percentages.copy())
            top_scores = top_five_scores(sorted_percentages)
            print("Top five scores:", top_scores)

        elif choice == 3:
            print("Exiting program.")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:


File Edit Selection View Go Run Terminal Help
yash
EXPLORER
YASH
.vscode
myboy1.py
myboy2.py
myboy3.py
myboy4.py
myboy5.py
myboy6.py
myboy1.py
myboy2.py
myboy3.py
myboy4.py
myboy5.py
myboy6.py
myboy6.py > main
15 def main():
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ValueError: invalid literal for int() with base 10: '& C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Admin/yash/myboy3.py'
PS C:\Users\Admin\yash> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Admin/yash/myboy6.py
Enter the number of students: 4
Enter percentage: 56
Enter percentage: 78
Enter percentage: 89
Enter percentage: 90
--- Menu ---
1. Sort using Quick Sort
2. Display Top Five Scores
3. Exit
Enter your choice: 1
Sorted array using Quick Sort: [56.0, 78.0, 89.0, 90.0]
--- Menu ---
1. Sort using Quick Sort
2. Display Top Five Scores
3. Exit
Enter your choice: 2
Top five scores: [56.0, 78.0, 89.0, 90.0]
--- Menu ---
1. Sort using Quick Sort
2. Display Top Five Scores
3. Exit
Enter your choice:
Python
Python Deb...
Python
OUTLINE

Practical no. 7

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:
Department of Computer Engineering has student's club named 'Pinnacle Club'.
Students of second, third and final year of department can be granted membership
on request. Similarly one may cancel the membership of club. First node is
reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked
list. Store student PRN and Name. Write functions to:
a) Add and delete the members as well as president or even secretary.
b) Compute total number of members of club
c) Display members
d) Two linked lists exist for two divisions. Concatenate two lists.

CODE:

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

class Node {
```

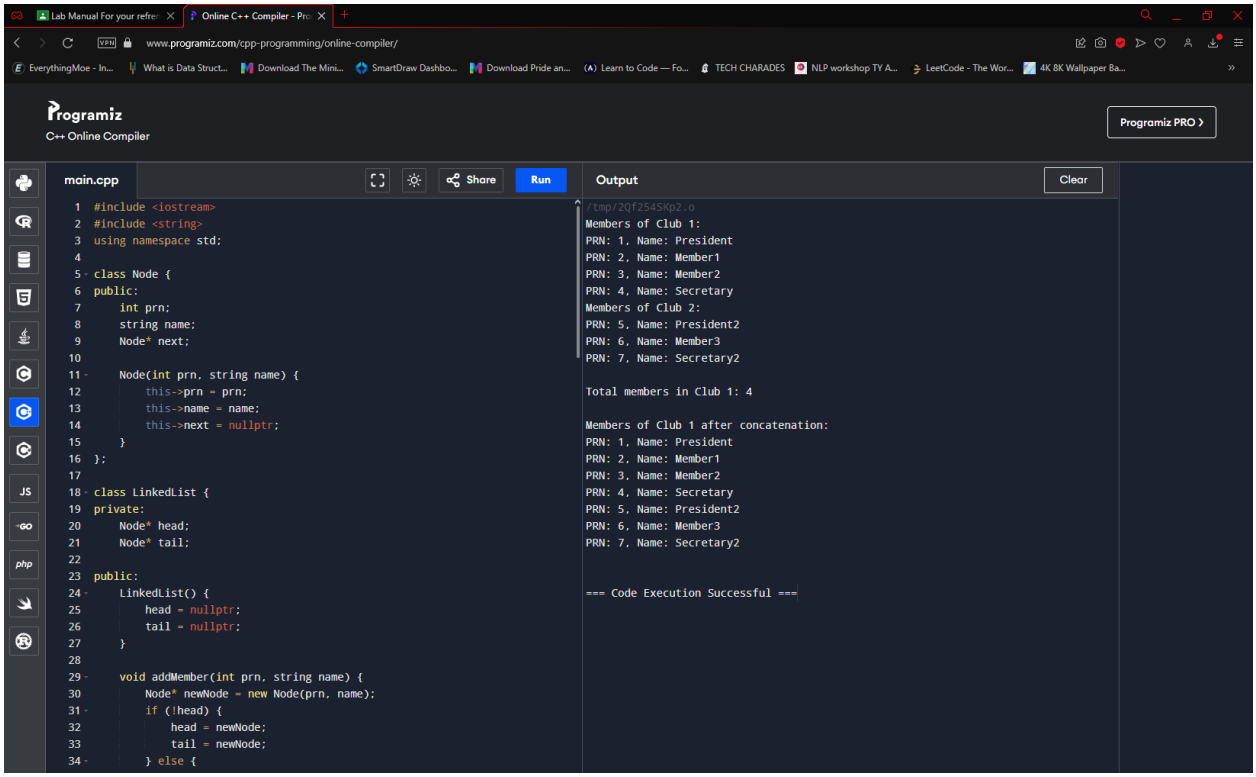
```
public:
    int prn;
    string name;
    Node* next;

    Node(int prn, string name) {
        this->prn = prn;
        this->name = name;
        this->next = nullptr;
    }
};

class LinkedList {
private:
    Node* head;
    Node* tail;

public:
    LinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void addMember(int prn, string name) {
        Node* newNode = new Node(prn, name);
        if (!head) {
            head = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }
    }

    void addPresident(int prn, string name) {
        Node* newNode = new Node(prn, name);
        if (!head) {
            head = newNode;
            tail = newNode;
        } else {
            newNode->next = head;
            head = newNode;
        }
    }

    void addSecretary(int prn, string name) {
```

```
    Node* newNode = new Node(prn, name);
    if (!head) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
}

void deleteMember(int prn) {


if (!head) return;

    if (head->prn == prn) {
        Node* temp = head;
        head = head->next;
        delete temp;
        if (!head) tail = nullptr;
        return;
    }

    Node* current = head;
    while (current->next && current->next->prn != prn) {
        current = current->next;
    }

    if (current->next) {
        Node* temp = current->next;
        current->next = current->next->next;
        if (temp == tail) tail = current;
        delete temp;
    }
}

int totalMembers() {
    int count = 0;
    Node* current = head;
    while (current) {
        count++;
        current = current->next;
    }
    return count;
}
```

```
    void displayMembers() {
        Node* current = head;
        while (current) {
            cout << "PRN: " << current->prn << ", Name: " << current->name <<
endl;
            current = current->next;
        }
    }

    void concatenate(LinkedList& other) {
        if (!head) {


     head = other.head;
            tail = other.tail;
        } else if (other.head) {
            tail->next = other.head;
            tail = other.tail;
        }
    }
};

int main() {
    LinkedList club1, club2;

    club1.addPresident(1, "President");
    cl
ub1.addMember(2, "Member1");
    club1.addMember(3, "Member2");
    club1.addSecretary(4, "Secretary");

    club2.addPresident(5, "President2");
    club2.addMember(6, "Member3");
    club2.addSecretary(7, "Secretary2");

    cout << "Members of Club 1:" << endl;
    club1.displayMembers();

    cout << "Members of Club 2:" << endl;
    club2.displayMembers();

    cout << "\nTotal members in Club 1: " << club1.totalMembers() <<
endl;

    club1.concatenate(club2);
```

```
    cout << "\nMembers of Club 1 after concatenation:" << endl;
    club1.displayMembers();

    return 0;
}
```

OUTPUT:



Practical no. 8

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream. Write C++ program to

store two sets using linked list. compute and display-
a) Set of students who like both vanilla and butterscotch

b) Set of students who like either vanilla or butterscotch or not both
c) Number of students who like neither vanilla nor butterscotch

CODE:

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

class Node {
public:
   string name;
   Node* next;
```

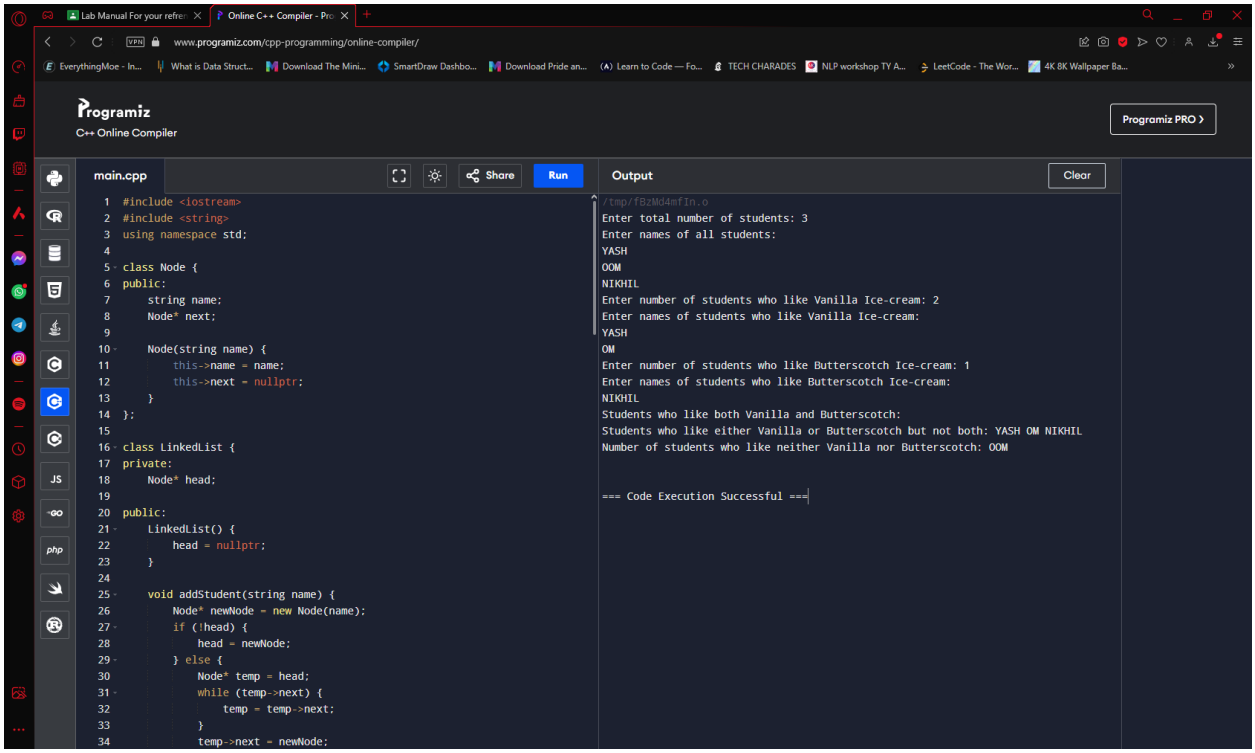
```
    Node(string name) {
        this->name = name;
        this->next = nullptr;
    }
};

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {


 head = nullptr;
    }

    void addStudent(string name) {
        Node* newNode = new Node(name);
        if (!head) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    bool searchStudent(string name) {
        Node* temp = head;
        while (temp) {
            if (temp->name == name) {
                return true;
            }
            temp = temp->next;
        }
        return false;
    }

    void displayStudents() {
        Node* temp = head;
        while (temp) {
            cout << temp->name << " ";
            temp = temp->next;
```

```
        }
        cout << endl;
    }

    Node* getHead() {
        return head;
    }
};

void computeBoth(LinkedList& listA, LinkedList& listB, LinkedList& both) {
    Node* temp = listA.getHead();
    while (temp) {



   if (listB.searchStudent(temp->name)) {
            both.addStudent(temp->name);
        }
        temp = temp->next;
    }
}

void computeEitherButNotBoth(LinkedList& listA, LinkedList& listB, LinkedList&
either) {
    Node* temp = listA.getHead();
    while (temp) {
        if (!listB.searchStudent(temp->name)) {
            either.addStudent(temp->name);
        }
        temp = temp->next;
    }
    temp = listB.getHead();
    while (temp) {
        if (!listA.searchStudent(temp->name)) {
            either.addStudent(temp->name);
        }
        temp = temp->next;
    }
}

void computeNeither(LinkedList& total, LinkedList& either, LinkedList& neither) {
    Node* temp = total.getHead();
    while (temp) {
        if (!either.searchStudent(temp->name)) {
            neither.addStudent(temp->name);
        }
```

```
        temp = temp->next;
    }
}

int main() {
    LinkedList setA, setB, both, either, neither, total;
    int n;
    string name;

    // Input total students
    cout << "Enter total number of students: ";
    cin >> n;

    cout << "Enter names of all students: " << endl;
    for (int i = 0; i < n; i++) {
        cin >> name;
        total.addStudent(name);
    }

    // Input Set A students
    cout << "Enter number of students who like Vanilla Ice-cream: ";
    cin >> n;

    cout << "Enter names of students who like Vanilla Ice-cream: " << endl;
    for (int i = 0; i < n; i++) {
        cin >> name;
        setA.addStudent(name);
    }

    // Input Set B students
    cout << "Enter number of students who like Butterscotch Ice-cream: ";
    cin >> n;

    cout << "Enter names of students who like Butterscotch Ice-cream: " << endl;
    for (int i = 0; i < n; i++) {
        cin >> name;
        setB.addStudent(name);
    }

    // Compute required sets
    computeBoth(setA, setB, both);
    computeEitherButNotBoth(setA, setB, either);
    computeNeither(total, either, neither);

    // Display results
    cout << "Students who like both Vanilla and Butterscotch: ";
```

```
    both.displayStudents();

    cout << "Students who like either Vanilla or Butterscotch but not both: ";
    either.displayStudents();

    cout << "Number of students who like neither Vanilla nor Butterscotch: ";
    neither.displayStudents();

    return 0;
}
```

OUTPUT:



Practical no. 9

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

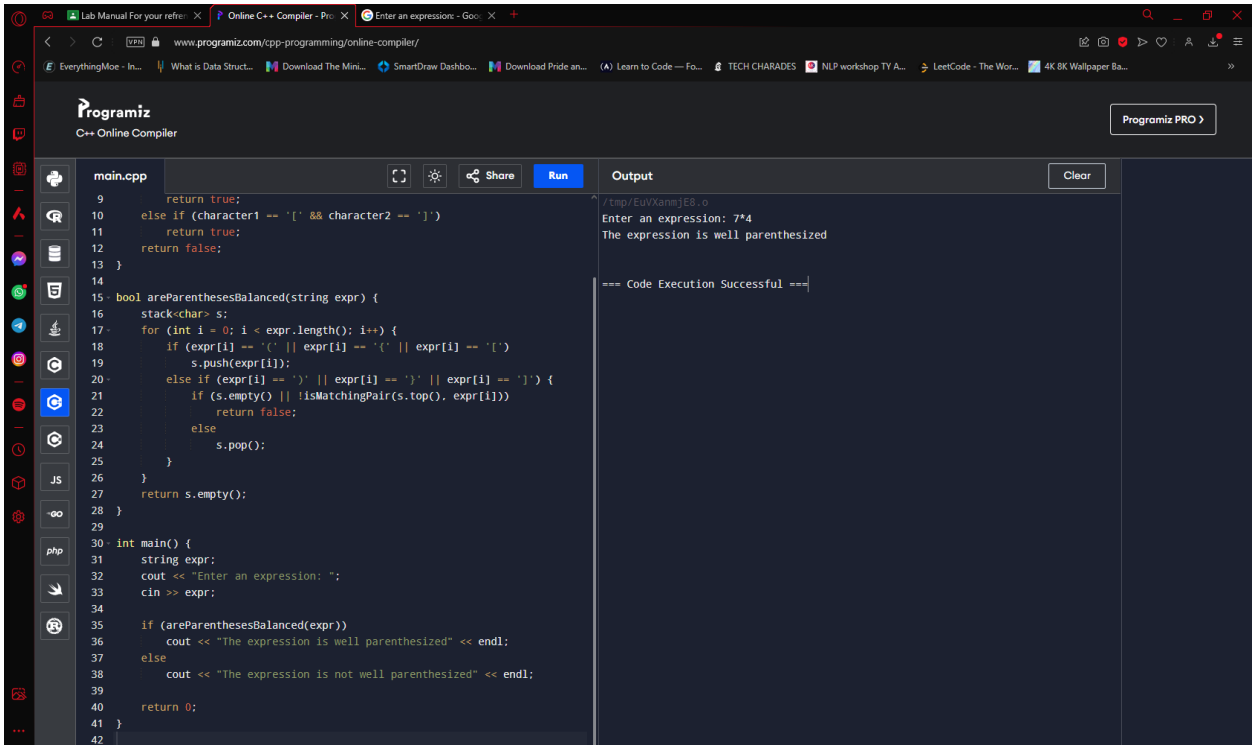
Problem Statement:
In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[]. Write C++ program using stack to check whether given
expression is well parenthesized or not.

CODE:

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

bool isMatchingPair(char character1, char character2) {
   if (character1 == '(' && character2 == ')')
      return true;
   else if (character1 == '{' && character2 == '}')
      return true;
   else if (character1 == '[' && character2 == ']')
      return true;
   return false;
}

bool areParenthesesBalanced(string expr) {
   stack<char> s;
   for (int i = 0; i < expr.length(); i++) {
```

```
        if (expr[i] == '(' || expr[i] == '{' || expr[i] == '[')
            s.push(expr[i]);
        else if (expr[i] == ')' || expr[i] == '}' || expr[i] == ']') {
            if (s.empty() || !isMatchingPair(s.top(), expr[i]))
                return false;
            else
                s.pop();
        }
    }
    return s.empty();
}

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

    if (areParenthesesBalanced(expr))
        cout << "The expression is well parenthesized" << endl;
    else
        cout << "The expression is not well parenthesized" << endl;

    return 0;
}
```

OUTPUT:



Practical no. 10

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

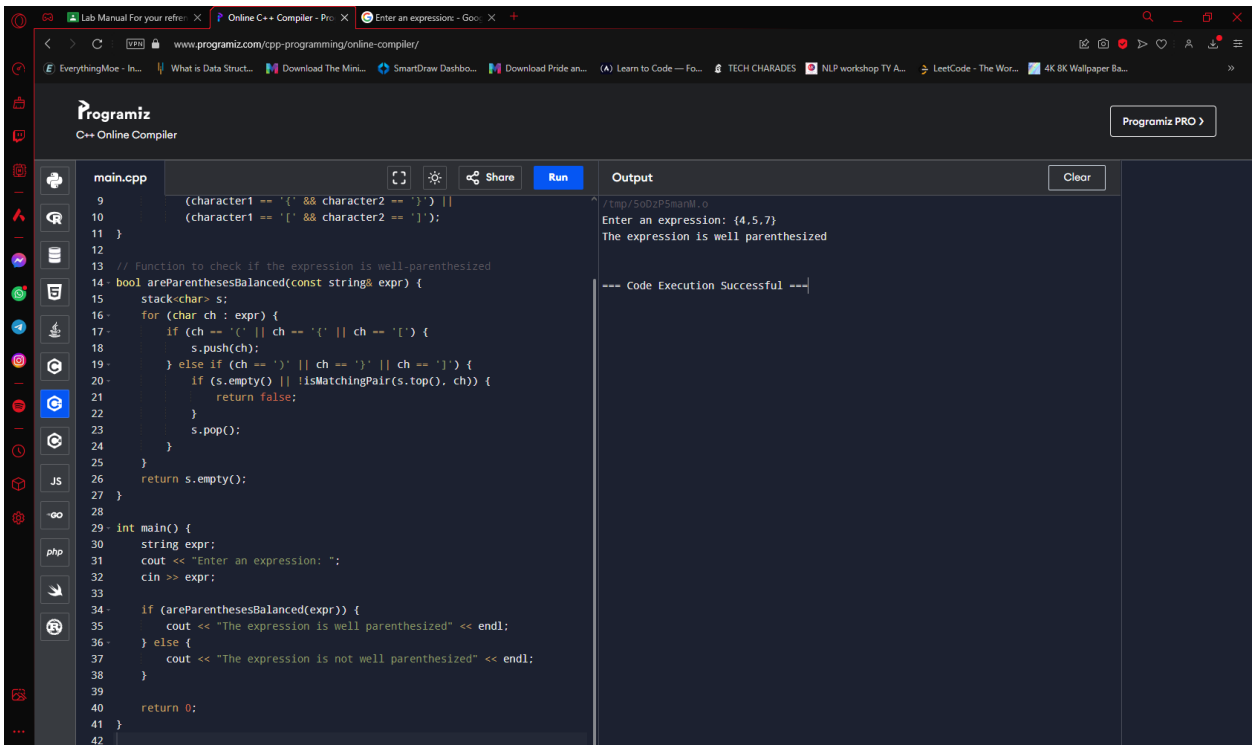
In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[]. Write C++ program using stack to check whether given

expression is well parenthesized or not.

CODE:

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

// Function to check if pairs match
bool isMatchingPair(char character1, char character2) {
    return (character1 == '(' && character2 == ')') ||
           (character1 == '{' && character2 == '}') ||
           (character1 == '[' && character2 == ']');
}

// Function to check if the expression is well-parenthesized
bool areParenthesesBalanced(const string& expr) {
    stack<char> s;
    for (char ch : expr) {
        if (ch == '(' || ch == '{' || ch == '[') {
            s.push(ch);
        } else if (ch == ')' || ch == '}' || ch == ']') {
            if (s.empty() || !isMatchingPair(s.top(), ch)) {
                return false;
            }
            s.pop();
        }
    }
    return s.empty();
}

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

    if (areParenthesesBalanced(expr)) {
        cout << "The expression is well parenthesized" << endl;
    } else {
        cout << "The expression is not well parenthesized" << endl;
    }

    return 0;
}
```

OUTPUT:



Practical no. 11

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job and

delete job from queue.

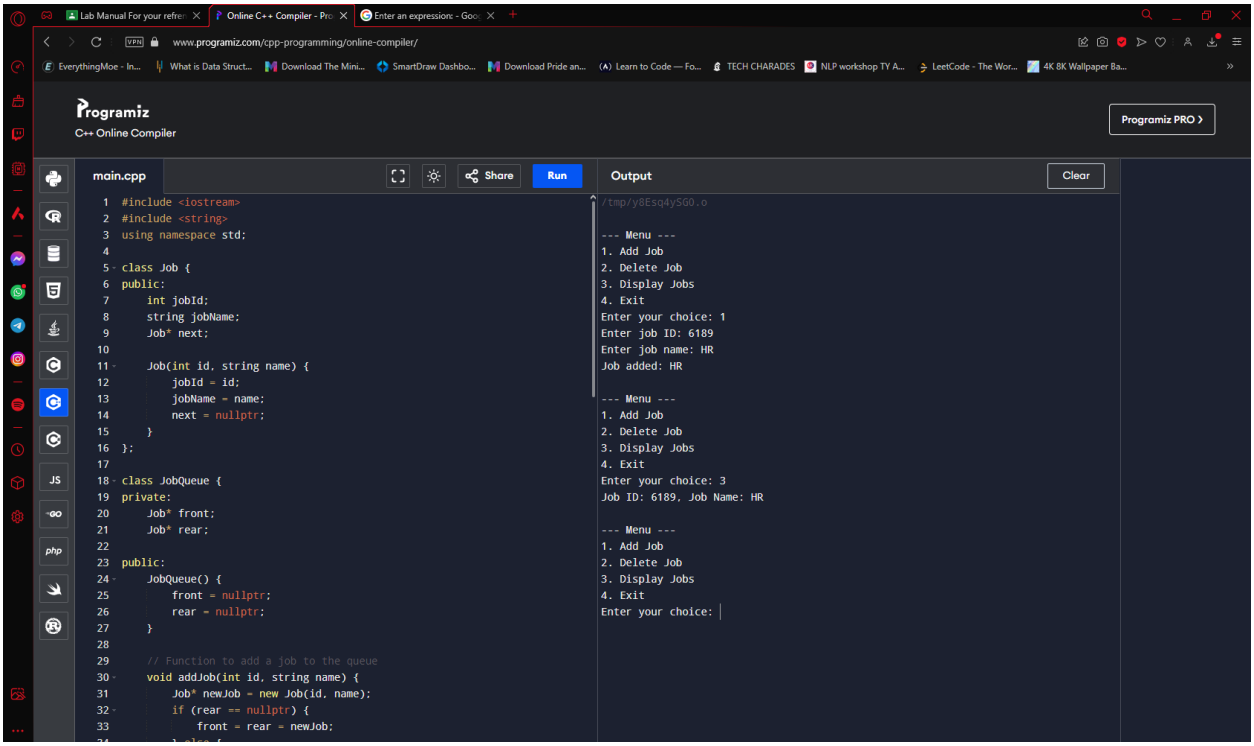
CODE:

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

class Job {
public:
   int jobId;
   string jobName;
   Job* next;

   Job(int id, string name) {
      jobId = id;
      jobName = name;
      next = nullptr;
   }
};

class JobQueue {
private:
   Job* front;
   Job* rear;

public:
   JobQueue() {
      front = nullptr;


     rear = nullptr;
   }

   // Function to add a job to the queue
   void addJob(int id, string name) {
      Job* newJob = new Job(id, name);
      if (rear == nullptr) {
         front = rear = newJob;
      } else {
         rear->next = newJob;
         rear = newJob;
      }
      cout << "Job added: " << name << endl;
   }

   // Function to delete a job from the queue
```

```
    void deleteJob() {
        if (front == nullptr) {
            cout << "No jobs to delete" << endl;
            return;
        }
        Job* temp = front;
        front = front->next;
        if (front == nullptr) {
            rear = nullptr;
        }
        cout << "Job deleted: " << temp->jobName << endl;
        delete temp;
    }

    // Function to display the jobs in the queue
    void displayJobs() {
        if (front == nullptr) {
            cout << "No jobs in the queue" << endl;
            return;
        }
        Job* temp = front;
        while (temp != nullptr) {
            cout << "Job ID: " << temp->jobId << ", Job Name: " << temp->jobName <<
endl;
            temp = temp->next;
        }
    }
};

int main() {
    JobQueue jobQueue;
    int choice, id;
    string name;

    while (true) {
        cout << "\n--- Menu ---" << endl;
        cout << "1. Add Job" << endl;
        cout << "2. Delete Job" << endl;
        cout << "3. Display Jobs" << endl;
        cout << "4. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter job ID: ";
```

```
            cin >> id;
            cout << "Enter job name: ";
            cin >> name;
            jobQueue.addJob(id, name);
            break;
        case 2:
            jobQueue.deleteJob();
            break;
        case 3:
            jobQueue.displayJobs();
            break;
        case 4:
            cout << "Exiting program." << endl;

            return 0;
        default:
            cout << "Invalid choice. Please try again." << endl;
      }
    }

    return 0;
}
```

OUTPUT:



Practical no. 12

Roll No. : 33
Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one- dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.

CODE:

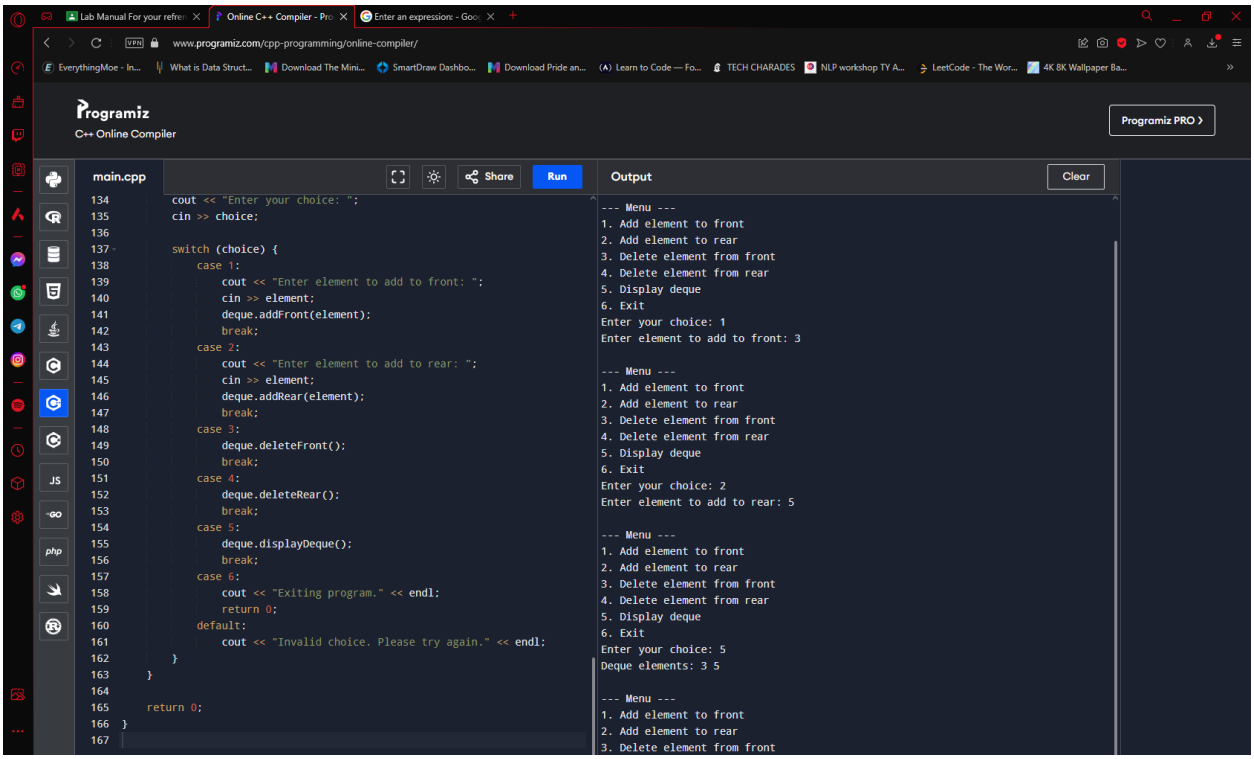
```
#include <iostream>
using namespace std;

class Deque {
private:
   int* arr;
   int front;
   int rear;
   int size;
   int capacity;

public:
   Deque(int capacity) {
      this->capacity = capacity;
      arr = new int[capacity];
      front = -1;
      rear = -1;
      size = 0;
   }

   ~Deque() {
      delete[] arr;
   }

   // Function to check if the deque is full
   bool isFull() {

 return size == capacity;
   }

   // Function to check if the deque is empty
   bool isEmpty() {
```

```
    return size == 0;
}

// Function to add an element to the front
void addFront(int element) {
    if (isFull()) {
        cout << "Deque is full. Cannot add element." << endl;
        return;
    }

    if (front == -1) {
        front = 0;
        rear = 0;
    } else {
        front = (front - 1 + capacity) % capacity;
    }

    arr[front] = element;
    size++;
}

// Function to add an element to the rear
void addRear(int element) {
    if (isFull()) {
        cout << "Deque is full. Cannot add element." << endl;
        return;
    }

    if (rear == -1) {
        front = 0;
        rear = 0;
    } else {
        rear = (rear + 1) % capacity;
    }

    arr[rear] = element;
    size++;
}

// Function to delete an element from the front

void deleteFront() {
    if (isEmpty()) {
        cout << "Deque is empty. Cannot delete element." << endl;
        return;
    }
```

```
        front = (front + 1) % capacity;
        size--;

        if (isEmpty()) {
            front = -1;
            rear = -1;
        }
    }

    // Function to delete an element from the rear
    void deleteRear() {
        if (isEmpty()) {
            cout << "Deque is empty. Cannot delete element." << endl;
            return;
        }

        rear = (rear - 1 + capacity) % capacity;
        size--;

        if (isEmpty()) {
            front = -1;
            rear = -1;
        }
    }

    // Function to display the elements of the deque
    void displayDeque() {
        if (isEmpty()) {
            cout << "Deque is empty." << endl;
            return;
        }

        cout << "Deque elements: ";
        for (int i = 0; i < size; i++) {
            cout << arr[(front + i) % capacity] << " ";
        }
        cout << endl;
    }
};

int main() {
    int capacity;
    cout << "Enter the capacity of the deque: ";
    cin >> capacity;
```

```
    Deque deque(capacity);
    int choice, element;

    while (true) {
        cout << "\n--- Menu ---" << endl;
        cout << "1. Add element to front" << endl;
        cout << "2. Add element to rear" << endl;
        cout << "3. Delete element from front" << endl;
        cout << "4. Delete element from rear" << endl;
        cout << "5. Display deque" << endl;
        cout << "6. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter element to add to front: ";
                cin >> element;
                deque.addFront(element);
                break;
            case 2:
                cout << "Enter element to add to rear: ";
                cin >> element;
                deque.addRear(element);
                break;
            case 3:
                deque.deleteFront();
                break;
            case 4:
                deque.deleteRear();
                break;
            case 5:
                deque.displayDeque();
                break;
            case 6:
                cout << "Exiting program." << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    }

    return 0;
}
```

OUTPUT:



Practical no. 12

Roll No. : 33

Name:Hegade Varunraj Bitu
Class: Second Year (Computer)
Division: A (Computer)

Problem Statement:

Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulate the system using circular queue using array.

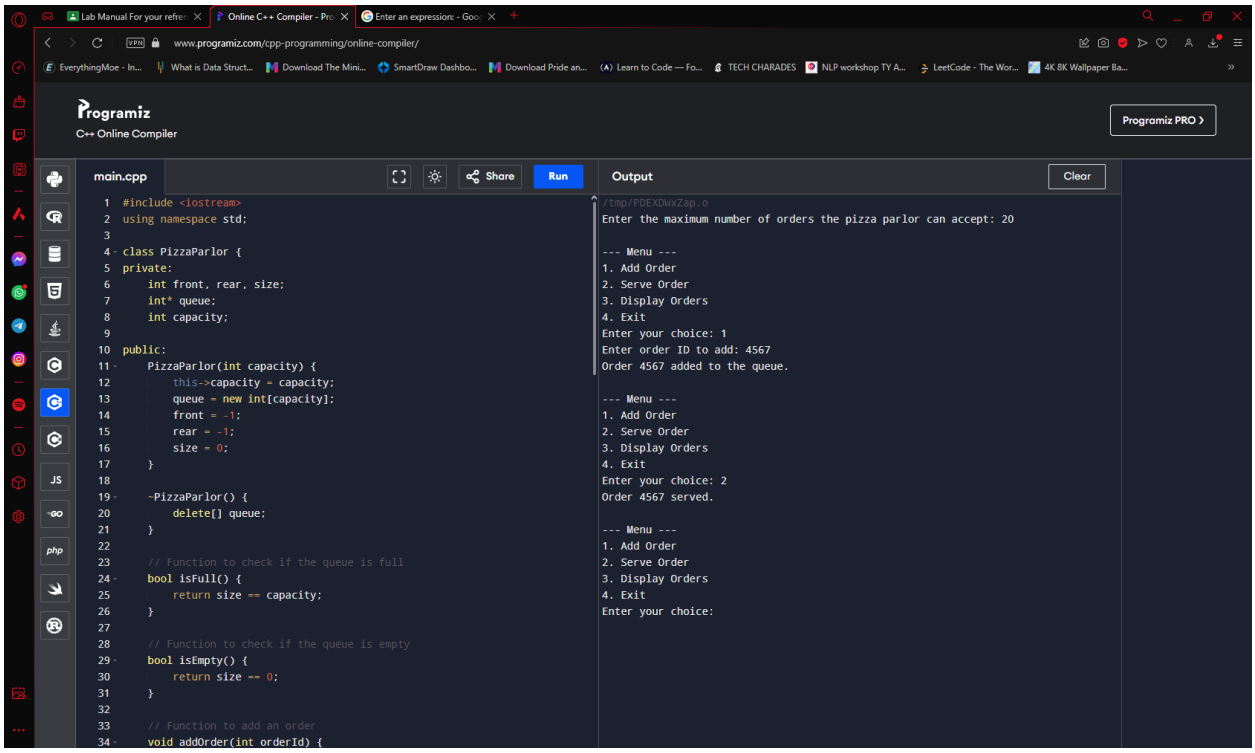
CODE:

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

class PizzaParlor {
private:
   int front, rear, size;
   int* queue;
   int capacity;

public:
   PizzaParlor(int capacity) {
      this->capacity = capacity;
      queue = new int[capacity];
      front = -1;
      rear = -1;
      size = 0;
   }

   ~PizzaParlor() {
      delete[] queue;
   }

   // Function to check if the queue is full
   bool isFull() {
      return size == capacity;
   }

   // Function to check if the queue is empty
   bool isEmpty() {
      return size == 0;
   }

   // Function to add an order
   void addOrder(int orderId) {
```

```
    if (isFull()) {
        cout << "Order queue is full. Cannot add more orders." << endl;
        return;
    }

    if (front == -1) {
        front = 0;
    }

    rear = (rear + 1) % capacity;
    queue[rear] = orderId;
    size++;
    cout << "Order " << orderId << " added to the queue." << endl;
}

// Function to serve an order
void serveOrder() {
    if (isEmpty()) {
        cout << "No orders to serve." << endl;
        return;
    }

    cout << "Order " << queue[front] << " served." << endl;
    front = (front + 1) % capacity;
    size--;

    if (isEmpty()) {
        front = -1;
        rear = -1;
    }
}

// Function to display the orders in the queue
void displayOrders() {
    if (isEmpty()) {
        cout << "No orders in the queue." << endl;
        return;
    }

    cout << "Orders in the queue: ";



 for (int i = 0; i < size; i++) {
        cout << queue[(front + i) % capacity] << " ";
    }
```

```
        cout << endl;
    }
};

int main() {
    int capacity;
    cout << "Enter the maximum number of orders the pizza parlor can accept: ";
    cin >> capacity;

    PizzaParlor pizzaParlor(capacity);
    int choice, orderId;

    while (true) {
        cout << "\n--- Menu ---" << endl;
        cout << "1. Add Order" << endl;
        cout << "2. Serve Order" << endl;
        cout << "3. Display Orders" << endl;
        cout << "4. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
cout << "Enter order ID to add: ";
                cin >> orderId;
                pizzaParlor.addOrder(orderId);
                break;
            case 2:
                pizzaParlor.serveOrder();
                break;
            case 3:
                pizzaParlor.displayOrders();
                break;
            case 4:
                cout << "Exiting program." << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    }

    return 0;
}
```

OUTPUT:


Lab Manual For your refrer
Online C++ Compiler - Pro
Enter an expression - Goo
www.programiz.com/cpp-programming/online-compiler/
Programiz
C++ Online Compiler
Programiz PRO ›
main.cpp
Share
Run
Output
Clear
#include <iostream>
using namespace std;
class PizzaParlor {
private:
int front, rear, size;
int* queue;
int capacity;
public:
PizzaParlor(int capacity) {
this->capacity = capacity;
queue = new int[capacity];
front = -1;
rear = -1;
size = 0;
}
~PizzaParlor() {
delete[] queue;
}
// Function to check if the queue is full
bool isFull() {
return size == capacity;
}
// Function to check if the queue is empty
bool isEmpty() {
return size == 0;
}
// Function to add an order
void addOrder(int orderId) {
/tmp/PDEXDWxZap.o
Enter the maximum number of orders the pizza parlor can accept: 20
--- Menu ---
1. Add Order
2. Serve Order
3. Display Orders
4. Exit
Enter your choice: 1
Enter order ID to add: 4567
Order 4567 added to the queue.
--- Menu ---
1. Add Order
2. Serve Order
3. Display Orders
4. Exit
Enter your choice: 2
Order 4567 served.
--- Menu ---
1. Add Order
2. Serve Order
3. Display Orders
4. Exit
Enter your choice: