Name :Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:01

TITLE: In second year computer engineering class, group A student's 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:

```
def union (a, b):
ans = a.copy()
   for i in b:
if i not in a:
        ans.append(i)
return ans def
minus (a, b):
   ans = []
for i in a:
      if i not in b:
ans.append(i)
return ans  def
intersection (a, b):
   ans = []
for i in a:
      if i in b: # Fix: should check if 'i' is in 'b'
ans.append(i)      return ans u = input ("ENTER THE
LIST: "). split (",")  b = input ("ENTER THE
BADMINTON LIST: "). split (",")
 # Fixed capitalization
```

```
c = input ("ENTER THE CRICKET LIST: "). split (",")  f = input ("ENTER THE
FOOTBALL LIST: "). split (",") print ("List of students who play CRICKET
and BADMINTON:", intersection
(b, c))
print ("List of the students who play either CRICKET or BADMINTON but not
BOTH:", union (minus (c, intersection (b, c)), minus (b, intersection (b, c))))
print ("Number of students who play neither CRICKET nor BADMINTON:",
minus (minus (u, b), c))
# Fixed 'a' to 'u'
print ("Number of students who play CRICKET and FOOTBALL but not
BADMINTON:", minus (intersection (c, f), b))
 # Fixed 'intersection' to 'intersection'
```

OUTPUT:

```
ENTER THE LIST: Tanvi, Arpita, Mayuri, Tanmaye
ENTER THE BADMINTON LIST: Tanvi, Arpita
ENTER THE CRICKET LIST: Mayuri, Arpita
ENTER THE FOOTBALL LIST: Tanvi, Tanmay
List of students who play CRICKET and BADMINTON: ['Arpita']
List of the students who play either CRICKET or BADMINTON but not BOTH:
['Mayuri', 'Tanvi']
Number of students who play neither CRICKET nor BADMINTON: ['Tanmay']
Number of students who play CRICKET and FOOTBALL but not BADMINTON:
```

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:02

TITLE: 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:

```
n = int(input("Enter the number of students: ")) present,
total, min_score, max_score, absent = 0, 0, 51, -1, 0  marks
= []  for i in range(n):
   temp = input(f"Enter the marks of student {i + 1} or AB if absent: ")
marks.append(temp)
   if temp != "AB":
present += 1         score
= int(temp)         total
+= score


      if score > max_score:
max_score   =   score   if
score < min_score:
min_score = score
   else:
      absent += 1
# Calculate average score avg = total /
present if present > 0 else 0
# Frequency calculation
frequency = {} for mark
in marks:
   if mark != "AB":
```

```
    mark_int = int(mark)        if
mark_int in frequency:
frequency[mark_int] += 1
    else:
      frequency[mark_int] = 1 #
Determine highest frequency
max_freq = 0
most_frequent_marks = [] for
mark, freq in frequency.items():
if freq > max_freq:        max_freq =
freq
    most_frequent_marks = [mark]
elif freq == max_freq:
    most_frequent_marks.append(mark)
# Print results  print(f"The average score of the class: {avg}")
print(f"Higher and lower score of class: {max_score} and {min_score}")
print(f"Students absent for test: {absent}")  print(f"Display marks
with highest frequency {max_freq}:
{most_frequent_marks}")
```

OUTPUT:

```
Enter the number of students: 6
Enter the marks of student 1 or AB if absent: 51
Enter the marks of student 2 or AB if absent: 66
Enter the marks of student 3 or AB if absent: AB
Enter the marks of student 4 or AB if absent: 77
Enter the marks of student 5 or AB if absent: 88
Enter the marks of student 6 or AB if absent: 99
The average score of the class: 76.2
Higher and lower score of class: 99 and 51
Students absent for test: 1
Display marks with highest frequency 1: [51, 66, 77, 88, 99]
```

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:03

TITLE: Write a Python program for department library which has N books, write functions for following: a) Delete the duplicate entries b) Display books in ascending order based on cost of books c) Count number of books with cost more than 500. d) Copy books in a new list which has cost less than 500.

CODE:

```
def delet(a):     ans = []
seen = set()     for item in
a:         if item[0] not in
seen:
ans.append(item)
seen.add(item[0])
return ans


def count(a):
   f = 0     less = []
for i in a:         if
int(i[1]) > 500:
         f += 1
else:
         less.append(i)
return f, less


def sort(a):
   ans = a.copy()  # Use parentheses to call the method
for i in range(len(ans)):
      for j in range(0, len(ans) - i - 1):
         if int(ans[j][1]) > int(ans[j + 1][1]): # Convert cost to
int for comparison
```

```
            ans[j], ans[j + 1] = ans[j + 1], ans[j]
return ans
n = int(input("ENTER THE NO OF BOOKS: "))
books = []
for i in range(n):
   a = input("ENTER THE ID OF BOOK " + str(i + 1) + ": ")     b =
input("ENTER THE COST OF BOOK " + str(i + 1) + ": ") books.append([a,
b]) print("ORIGINAL LIST:", books) books_no_duplicates = delet(books)
print("DELETE THE DUPLICATE ENTRIES:", books_no_duplicates)
sorted_books = sort(books_no_duplicates) print("DISPLAY BOOK IN
ASSCENDING ORDER BASED ON COST OF BOOK:",
sorted_books) # Use sorted_books instead of tup[0] tup =
count(books_no_duplicates) # Use books_no_duplicates instead of
book_no_duplicates print("COUNT NO OF BOOKS WITH COST MORE THAN
500:", tup[0]) print("COPY BOOKS IN A NEW LIST WHICH HAS COST LESS
THAN 500:", tup[1])
```

OUTPUT:

```
ENTER THE NO OF BOOKS: 5
ENTER THE ID OF BOOK 1: 1
ENTER THE COST OF BOOK 1: 500
ENTER THE ID OF BOOK 2: 2
ENTER THE COST OF BOOK 2: 600
ENTER THE ID OF BOOK 3: 3
ENTER THE COST OF BOOK 3: 900
ENTER THE ID OF BOOK 4: 4
ENTER THE COST OF BOOK 4: 100
ENTER THE ID OF BOOK 5: 2
ENTER THE COST OF BOOK 5: 600
ORIGINAL LIST: [['1', '500'], ['2', '600'], ['3', '900'], ['4', '100'],
['2', '600']]
DELETE THE DUPLICATE ENTRIES: [['1', '500'], ['2', '600'], ['3', '900'],
```

['4', '100']]

DISPLAY BOOK IN ASSCENDING ORDER BASED ON COST OF BOOK: [['4', '100'],

['1', '500'], ['2', '600'], ['3', '900']]

COUNT NO OF BOOKS WITH COST MORE THAN 500: 2

COPY BOOKS IN A NEW LIST WHICH HAS COST LESS THAN 500: [['1', '500'],

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:04

TITLE: Write a pythonprogram 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:

```
def selection_sort(arr):
    """Sort the array using the selection sort algorithm."""
n = len(arr)      for i in range(n):
        min_idx = i         for j in
range(i + 1, n):             if
arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
return arr
def bubble_sort(arr):
    """Sort the array using the bubble sort algorithm."""
    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  def display_top_five(arr):
    """Display the top five scores."""      print("Top five scores:")
for score in sorted(arr, reverse=True)[:5]: # Sort in descending
order and slice the top five
        print(f"{score:.2f}")
percentages = [75, 82.3, 91, 89.88, 53, 63, 85, 95, 99]
# Sorting using selection sort  stored_percentages_selection =
selection_sort(percentages[:])  print("Selection Sort:")
display_top_five(stored_percentages_selection)
```

```
# Sorting using bubble sort  stored_percentages_bubble =
bubble_sort(percentages[:])  print("\nBubble Sort:")
display_top_five(stored_percentages_bubble)
```

OUTPUT:

Selection Sort:

Top five scores:

99.00

95.00

91.00

89.88

85.00

Bubble Sort:

Top five scores:

99.00

95.00

91.00

89.88

85.00

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:05

TITLE: 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 to perform 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 to perform 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: # Corrected condition

            arr[j] = arr[j - gap]

            j -= gap

         arr[j] = temp  # This line was incorrect before
      gap //= 2  # Fixed indentation

return arr  # Added return statement
```

```
# Function to display top 5 scores  def
display_top_five(arr):
   top_five = sorted(arr, reverse=True)[:5] # Sort and take top 5
directly     print("Top 5 scores:")     for score in top_five:
      print(score)
# Example usage  if
__name__== "__main__":
   m = int(input("Enter total number of marks: "))      percentage
= []     for k in range(m):  # Changed to range(m) to include m
entries         n = float(input("Enter the marks: "))
percentage.append(n)      print("Original percentages:",
percentage)      # Sorting using insertion sort
sorted_per_insertion = insertion_sort(percentage.copy())
print("Sorted percentages using insertion sort:",
sorted_per_insertion)      display_top_five(sorted_per_insertion)
   # Sorting using shell sort      sorted_per_shell =
shell_sort(percentage.copy())      print("Sorted percentages
using shell sort:", sorted_per_shell)
display_top_five(sorted_per_shell)
```

OUTPUT:

```
Enter total number of marks: 7
Enter the marks: 66
Enter the marks: 77
Enter the marks: 88
Enter the marks: 55
Enter the marks: 99
Enter the marks: 87
Enter the marks: 76
Original percentages: [66.0, 77.0, 88.0, 55.0, 99.0, 87.0, 76.0]
Sorted percentages using insertion sort: [55.0, 66.0, 76.0, 77.0, 87.0, 88.0, 99.0]
Top 5 scores:
```

99.0

88.0

87.0

77.0

76.0

Sorted percentages using shell sort: [55.0, 66.0, 76.0, 77.0, 87.0, 88.0, 99.0]

Top 5 scores:

99.0

88.0

87.0

77.0 76.0

Name: Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:06

TITLE: 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:

```
#def quicksort(arr): if
len(arr)<=1:
return arr
else:
pivot=arr[0]  left=[x for x in arr[1:] if x<pivot]  middle=[x
for x in arr if x==pivot]  right=[x for x in arr[1:] if
x>=pivot]  return
quicksort(left)+middle+quicksort(right) m=int(input("
ENTER THE  number of PERCENTAGE : "))
percentage=[]  for i in range(m):
n=int(input(" ENTER THE PERCENTAGE : "))
percentage.append(n)
sorted_perecntage=quicksort(percentage)
print(" PERCENTAGE IN ASENDING ORDER : ")
top_five_per=sorted_perecntage[-5:]  print(" TOP
FIVE PERCENTAGE IS : ",top_five_per)
top_five_per.reverse()
print(" PERCENTAGE IN DESENDING ORDER : ",top_five_per)
```

OUTPUT:

```
ENTER THE  number of PERCENTAGE : 5
```

ENTER THE PERCENTAGE : 99
ENTER THE PERCENTAGE : 66

ENTER THE PERCENTAGE : 77

ENTER THE PERCENTAGE : 88

ENTER THE PERCENTAGE : 45

PERCENTAGE IN ASENDING ORDER : [99, 66, 77, 88, 45]

TOP FIVE PERCENTAGE IS : [45, 66, 77, 88, 99]

PERCENTAGE IN DESENDING ORDER : [99, 88, 77, 66, 45]

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:07

TITLE: 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 exists for two divisions. Concatenate two lists.

CODE:

```
#include<iostream>
#include<cstring> using
namespace std;
struct node{
        int prn;
char name[20];
node *next;
};
class pinnacle{
        node *head=NULL,*tail=NULL; public:
        void create(){
                int data;
char name1[20];
                node *temp=new node;

                cout<<"Enter the PRN"<<endl;
                cin>>data;
                cout<<"Enter the Name"<<endl;
        cin>>name1;             temp-
>prn=data;              strcpy(temp->name,
name1);                 temp->next=NULL;
if(head==NULL){
                        head=temp;
                        tail=temp;
                }
                else{
                        tail->next=temp;
                        tail=temp;
                }
        }
        void president(){
        int data;
```

```
                char name1[20]; node
                *temp=new node;
                cout<<"Enter the
                PRN"<<endl; cin>>data;
                cout<<"Enter the Name"<<endl; cin>>name1;
                temp->prn=data;         strcpy(temp-
        >name, name1);
                temp->next=head;
                head=temp;
        }
        void member(){
                int data,pos;           node
*mem;           char name1[20];
node *temp=new node;
cout<<"Enter the PRN"<<endl;
                cin>>data;
                cout<<"Enter the Name"<<endl;
                cin>>name1;             temp-
>prn=data;
                strcpy(temp->name, name1);
temp->next=NULL;                if(head==NULL){
                        head=temp;
                        tail=temp;
                }
                else{
                        cout<<"Enter the PRN after which you want to insert:"<<endl;
                        cin>>pos;
        mem=head;
                        while(mem!=NULL){
if(mem->prn==pos){                                      temp-
>next=mem->next;                                        mem-
>next=temp;                                     if(temp-
>next==NULL){
                                                        tail=temp;
                                        }
                                        break;
                                }
                                mem=mem->next;
                        }
                }
        }
        void display(){         node
        *print=head;
        while(print!=NULL){
                        cout<<print->prn<<"--->"<<print->name<<"\t";
                        print=print->next;
                }
                cout<<endl;
        }
        void total(){ int
                count=0;
```

```
		node *current=head; while(current!=NULL){
			count++;
			current=current->next;
		}
		cout<<"Total number of nodes:"<<count<<endl;
	}
	void Delete_pre(){
		if (head!=NULL){
 node *temp=head;  head= head-
>next;  delete temp;
		}else{
			cout<<"LIST IS EMPTY ";
		}

	}
	void Delete_sec(){
		if(head!=NULL){
	node *current=head;
node *prev=NULL;
			while(current->next !=NULL){
				prev=current;
				current=current->next;
			}
			if(prev !=NULL){
	prev->next=NULL;
				tail=prev;
			}
			else{
	head=NULL;
tail=NULL;
			}
			delete current;
		}
		else{
			cout<<"List is Empty";
		}

	}
	void Delete_mem(){
		int pos;
		cout<<"Enter the PRN after which you want to delete";
		cin>>pos;
		node *current=head;
	node *prev=NULL;
		while(current!=NULL && current->prn!=pos){
			prev=current;
		current=current->next;
		}
		if(current !=NULL && current->next!=NULL){
			node *temp=current->next;
		current->next=temp->next;	if(temp==tail){
```

```
                tail=current;
            }
            delete temp;
        }
        else{
            cout<<"Position not found or no member after the given PRN";
        }

    }
};


int main(){
 int ch1, ch2, ch3; char
ch;  pinnacle p;
    do{
        cout<<"\t_____MENU_____"<<endl;
cout<<"\t1.CREATE"<<endl;
cout<<"\t2.DISPLAY"<<endl;          cout<<"\t3.INSERT
MEMBER"<<endl;          cout<<"\t4.TOTAL
MEMBERS"<<endl;          cout<<"\t5.DELET"<<endl;
        cout<<"****************************************"<<endl;
        cout<<"\tENTER YOUR CHOICE : "<<endl;
        cin>>ch1;
        switch(ch1){
    case 1:
            p.create();
        break;
case 2:
            p.display();
            break;
    case 3:
            cout<<"****************************************"<<endl;
        cout<<"\t1.PRESIDENT"<<endl;
cout<<"\t2.MEMBERS"<<endl;                                    cout<<"\t3.SECRATORY"<<endl;
            cout<<"\tENTER YOUR CHOICE : "<<endl;
            cin>>ch2;
        switch(ch2){
    case  1:
                                p.president();
                                break;
        case 2:
                                p.member();
                                break;
            case 3:
                                p.create();
                                break;
            }
            break;
        case 4:
            p.total();
```

```
                    break;
                case 5:
                    cout<<"*****************************************"<<endl;
                    cout<<"\t1.PRESIDENT"<<endl;
                cout<<"\t2.MEMBERS"<<endl;
                cout<<"\t3.SECRATORY"<<endl;
                    cout<<"\tENTER YOUR CHOICE : "<<endl;
                    cin>>ch3;
                    switch(ch3){
                case  1:
                                    p.Delete_pre();
                                    break;
                case 2:
                                    p.Delete_mem();
                                    break;
                case 3:
                                    p.Delete_sec();
                                    break;
                    }
                    break;
        default :
                    cout<<"\t YOU HAVE ENTERED THE WRONG CHOICE"<<endl;

break;

            }
            cout<<" DO YOU WAN'T TO COTINUE (y/n): ";
            cin>>ch;
            }
            while(ch=='y' || ch=='Y');
            return 0;
}
```

OUTPUT:

```
          _________MENU_____________
     1.CREATE
     2.DISPLAY
     3.INSERT MEMBER
     4.TOTAL MEMBERS
     5.DELET
*****************************************
ENTER YOUR CHOICE :
1
Enter the PRN
1
Enter the Name
TANVI
 DO YOU WAN'T TO COTINUE (y/n): y
     _________MENU_____________
     1.CREATE
```

```
      2.DISPLAY
      3.INSERT MEMBER
      4.TOTAL MEMBERS
      5.DELET
*****************************************
ENTER YOUR CHOICE :
2
1--->TANVI
 DO YOU WAN'T TO COTINUE (y/n): y
      _________MENU_________
      1.CREATE
      2.DISPLAY
      3.INSERT MEMBER
      4.TOTAL MEMBERS
      5.DELET
*****************************************
ENTER YOUR CHOICE :
3
*****************************************
      1.PRESIDENT
      2.MEMBERS
      3.SECRATORY
ENTER YOUR CHOICE :
1
Enter the PRN
11
Enter the Name
ARPITA
 DO YOU WAN'T TO COTINUE (y/n): y
      _________MENU_________
      1.CREATE
      2.DISPLAY
      3.INSERT MEMBER
      4.TOTAL MEMBERS
      5.DELET
*****************************************
ENTER YOUR CHOICE :
3
*****************************************
      1.PRESIDENT
      2.MEMBERS
      3.SECRATORY
ENTER YOUR CHOICE :
2
Enter the PRN
12
Enter the Name
MAYURI
Enter the PRN after which you want to insert: 11
 DO YOU WAN'T TO COTINUE (y/n): y
      _________MENU_________
```

```
      1.CREATE
      2.DISPLAY
      3.INSERT MEMBER
      4.TOTAL MEMBERS
      5.DELET ****************************************        ENTER YOUR CHOICE :
3
****************************************
      1.PRESIDENT
      2.MEMBERS
      3.SECRATORY
ENTER YOUR CHOICE :
3
Enter the PRN
13
Enter the Name
TANMAYE
 DO YOU WAN'T TO COTINUE (y/n): y
      _________MENU_____________
      1.CREATE
      2.DISPLAY
      3.INSERT MEMBER
      4.TOTAL MEMBERS
      5.DELET
****************************************
ENTER YOUR CHOICE :
4
Total number of nodes:4
 DO YOU WAN'T TO COTINUE (y/n): y
      _________MENU_____________
      1.CREATE
      2.DISPLAY
      3.INSERT MEMBER
      4.TOTAL MEMBERS
      5.DELET
****************************************
ENTER YOUR CHOICE :
5
****************************************
      1.PRESIDENT
      2.MEMBERS
      3.SECRATORY
ENTER YOUR CHOICE :
1
 DO YOU WAN'T TO COTINUE (y/n): y
      _________MENU_____________
      1.CREATE
      2.DISPLAY
      3.INSERT MEMBER
      4.TOTAL MEMBERS
      5.DELET
```

```
****************************************
ENTER YOUR CHOICE :
2
12--->MAYURI   1--->TANVI     13--->TANMAYE
 DO YOU WAN'T TO COTINUE (y/n): n
```

Name :Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:08

TITLE: 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 <unordered_set>
#include <string>

using namespace std;

struct Node {
string name;
   Node* next;
};

// Function to insert a student into the linked list
void insert(Node*& head, const string& name) {
Node* newNode = new Node{name, nullptr};     if
(!head) {
      head = newNode;
   } else {
      Node* temp = head;
while (temp->next) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
}

// Function to display the linked list
void display(Node* head) {     while
(head) {
      cout << head->name << " ";
head = head->next;
   }
   cout << endl;
}

// Function to compute the intersection of two sets unordered_set<string>
intersection(Node* listA, Node* listB) {     unordered_set<string> setA;
   unordered_set<string> result;
```

```
    // Store names from listA while
    (listA) {        setA.insert(listA-
    >name);        listA = listA->next;
    }

    // Store names from listB and find intersection
while (listB) {
        if (setA.find(listB->name) != setA.end()) {            result.insert(listB-
>name);
        }
        listB = listB->next;
    }

    return result;
}

// Function to compute the union of two sets unordered_set<string>
unionSets(Node* listA, Node* listB) {     unordered_set<string>
result;

    // Add names from listA
while (listA) {
result.insert(listA->name);
listA = listA->next;
    }

    // Add names from listB
while (listB) {
result.insert(listB->name);
listB = listB->next;
    }

    return result;
}

int main() {
    Node* vanillaStudents = nullptr;
Node* butterscotchStudents = nullptr;
    int totalStudents;

    // Input total number of students     cout
<< "Enter total number of students: ";     cin
>> totalStudents;
    cin.ignore(); // To ignore the newline character after the integer input

    // Input names of all students
    cout << "Enter names of all students (separated by newlines):\n";
    for (int i = 0; i < totalStudents; ++i) {
string name;        getline(cin, name);
        // You can store these names in an array if needed
    }
```

```
   // Specify students who like vanilla int
   n;
   cout << "Enter number of students who like vanilla ice-cream: ";
cin >> n;
   cin.ignore(); // Ignore newline after integer input
   cout << "Enter names of students who like vanilla ice-cream:\n";
for (int i = 0; i < n; ++i) {        string name;        getline(cin, name);
      insert(vanillaStudents, name);
   }

   // Specify students who like butterscotch
   cout << "Enter number of students who like butterscotch ice-cream: ";
cin >> n;
   cin.ignore(); // Ignore newline after integer input
   cout << "Enter names of students who like butterscotch ice-cream:\n";
for (int i = 0; i < n; ++i) {        string name;        getline(cin, name);
      insert(butterscotchStudents, name);
   }

   // Compute intersection
   unordered_set<string> both = intersection(vanillaStudents, butterscotchStudents);
cout << "Students who like both vanilla and butterscotch: ";     for (const auto& name
: both) {        cout << name << " ";
   }
   cout << endl;

   // Compute union
   unordered_set<string> either = unionSets(vanillaStudents, butterscotchStudents);
cout << "Students who like either vanilla or butterscotch (or both): ";     for (const
auto& name : either) {
      cout << name << " ";
   }
   cout << endl;

   // Calculate and display number of students who like neither
   int neither = totalStudents - either.size();
   cout << "Number of students who like neither vanilla nor butterscotch: " << neither << endl;

   // Free linked list memory while
(vanillaStudents) { Node* temp =
vanillaStudents; vanillaStudents =
vanillaStudents->next;       delete temp;
   }
   while (butterscotchStudents) {
Node* temp = butterscotchStudents;
      butterscotchStudents = butterscotchStudents->next;
delete temp;
   }

   return 0;
```

}

OUTPUT:

```
Enter total number of students: 4
Enter names of all students (separated by newlines):
tanvi arpita tanmaye mayuri

Enter number of students who like vanilla ice-cream: 2

Enter names of students who like vanilla ice-cream:
tanvi arpita

Enter number of students who like butterscotch ice-cream: 1

Enter names of students who like butterscotch ice-cream:
tanvi

Students who like both vanilla and butterscotch: tanvi

Students who like either vanilla or butterscotch (or both): arpita tanvi

Number of students who like neither vanilla nor butterscotch: 2
```

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:09

TITLE: 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;

bool isBalanced(const std::string& expression) {
   stack<char> stack;

   for (char ch : expression) {
      // Push opening brackets onto the stack
if (ch == '(' || ch == '{' || ch == '[') {
stack.push(ch);
      }
      // Check for closing brackets        else
if (ch == ')' || ch == '}' || ch == ']') {
// If stack is empty, it's unbalanced
         if (stack.empty()) {
return false;
         }
         // Check for matching opening bracket
         char top = stack.top();
         stack.pop();           if ((ch
== ')' && top != '(') ||              (ch
== '}' && top != '{') ||              (ch
== ']' && top != '[')) {
return false;
         }
      }
   }

   // If the stack is empty, all brackets were matched
return stack.empty();
}

int main() {    string
expression;
   cout << "Enter an expression: ";
getline(std::cin, expression);

   if (isBalanced(expression)) {
      cout << " VALID EXPRESSION " << std::endl;
   } else {
```

```
        cout << " INVALID EXPRESSION " << std::endl;
    }

    return 0;
}
```

OUTPUT:

```
Enter an expression: ()
 VALID EXPRESSION

Enter an expression: ({})
 VALID EXPRESSION

 Enter an expression: {[}
 INVALID EXPRESSION
```

Name :Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:10

TITLE: Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions: 1. Operands and operator, both must be single character. 2. Input Postfix expression must be in a desired format. 3. Only '+', '-', '*' and '/ ' operators are expected.

CODE:

```
#include <iostream>
#include <stack>
#include <string>


using namespace std;

// Function to determine the precedence of operators
int precedence(char op) {     if (op == '+' || op == '-')
return 1;
   if (op == '*' || op == '/')
return 2;     return 0;
}

// Function to convert infix expression to postfix expression
string infixToPostfix(const string& infix) {     stack<char>
opStack;     string postfix;

   for (char ch : infix) {
      if (isalnum(ch)) {  // If the character is an operand
postfix += ch;        } else if (ch == '(') {
opStack.push(ch);        } else if (ch == ')') {
          while (!opStack.empty() && opStack.top() != '(') {
             postfix += opStack.top();
opStack.pop();
          }
          opStack.pop();  // Remove the '(' from stack
} else {  // The character is an operator
          while (!opStack.empty() && precedence(opStack.top()) >= precedence(ch)) {
postfix += opStack.top();                opStack.pop();
          }
          opStack.push(ch);
      }
   }

   // Pop all the operators from the stack
while (!opStack.empty()) {        postfix
+= opStack.top();
```

```
      opStack.pop();
   }

   return postfix;
}

// Function to evaluate the postfix expression int
evaluatePostfix(const string& postfix) {
stack<int> valueStack;

   for (char ch : postfix) {
if (isdigit(ch)) {
         valueStack.push(ch - '0');  // Convert char to int
} else {  // The character is an operator            int right
= valueStack.top(); valueStack.pop();           int  left  =
valueStack.top(); valueStack.pop();           switch (ch)
{              case '+': valueStack.push(left + right);
break;               case '-': valueStack.push(left - right);
break;               case '*': valueStack.push(left * right);
break;               case '/':                    if (right != 0)
                  valueStack.push(left / right);
else {
                  cout << "Error: Division by zero" << endl;
return 0;
               }
break;
         }
      }
   }

   return valueStack.top();  // The result is the last item on the stack
}

int main() {
   string infix;

   cout << "Enter an infix expression (e.g., A+B*C): ";
cin >> infix;

   // Convert infix to postfix     string postfix =
infixToPostfix(infix);    cout << "Postfix
expression: " << postfix << endl;

   // Evaluate the postfix expression (only single digit operands)
int result = evaluatePostfix(postfix);     cout << "Result of
evaluation: " << result << endl;

   return 0;
}
```

OUTPUT :

```
Enter an infix expression (e.g., A+B*C): A+B*C+D
Postfix expression: ABC*+D+
Segmentation fault (core dumped)
Enter an infix expression (e.g., A+B*C): 3+4*5+6
Postfix expression: 345*+6+
Result of evaluation: 29
```

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:11

TITLE: 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 <queue>
#include <string>

class JobQueue { private:
   std::queue<std::string> jobs;

public:
   // Function to add a job to the queue
void addJob(const std::string& job) {
      jobs.push(job);
      std::cout << "Job added: " << job << std::endl;
   }

   // Function to delete a job from the queue
   void deleteJob() {
if (jobs.empty()) {
         std::cout << "No jobs to delete." << std::endl;
return;
      }
      std::cout << "Job deleted: " << jobs.front() << std::endl;
jobs.pop();
   }

   // Function to display the current jobs in the queue
void displayJobs() const {        if (jobs.empty()) {
         std::cout << "The job queue is empty." << std::endl;
return;
      }

      std::cout << "Current jobs in the queue: ";
      std::queue<std::string> tempQueue = jobs; // Create a copy to display
while (!tempQueue.empty()) {           std::cout << tempQueue.front() << "
";           tempQueue.pop();
      }
      std::cout << std::endl;
   }
};
```

```
int main() {
JobQueue jobQueue;
int choice;
std::string job;

   do {
      std::cout << "\nJob Queue Menu:\n";
std::cout << "1. Add Job\n";        std::cout
<< "2. Delete Job\n";        std::cout << "3.
Display Jobs\n";        std::cout << "4.
Exit\n";        std::cout << "Enter your
choice: ";
      std::cin >> choice;

      switch (choice) {
case 1:
            std::cout << "Enter job name: ";
            std::cin >> job;
jobQueue.addJob(job);
break;           case 2:
            jobQueue.deleteJob();
break;           case 3:
            jobQueue.displayJobs();
break;           case 4:
            std::cout << "Exiting..." << std::endl;
            break;
default:
            std::cout << "Invalid choice, please try again." << std::endl;
      }
   } while (choice != 4);

   return 0;
}
```

OUTPUT:

Job Queue Menu:

//HERE WE ARE ADDING SOME JOBS
1. Add Job
2. Delete Job
3. Display Jobs 4. Exit
Enter your choice: 1
Enter job name: TEACHER
Job added: TEACHER

Job Queue Menu:

1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter your choice: 1
Enter job name: DOCTOR
Job added: DOCTOR

Job Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs 4. Exit
Enter your choice: 1
Enter job name: ENGINEER
Job added: ENGINEER

//DISPLAY THE ADDED JOBS Job
Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter your choice: 3
Current jobs in the queue: TEACHER DOCTOR ENGINEER

//DELETE THE 1ST JOB Job
Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter your choice: 2
Job deleted: TEACHER

Job Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter your choice: 3
Current jobs in the queue: DOCTOR ENGINEER

//DELETE THE 2ND JOB Job
Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs 4. Exit
Enter your choice: 2
Job deleted: DOCTOR

Job Queue Menu:

1. Add Job
2. Delete Job
3. Display Jobs 4. Exit
Enter your choice: 3
Current jobs in the queue: ENGINEER

//DELETE THE 3RD JOB Job
Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter your choice: 2
Job deleted: ENGINEER

//TOTAL JOBS ARE DELETED Job
Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit Enter your choice: 3 The job queue is empty. //(NOW YOUR QUEUE BECOME EMPTY)
Job Queue Menu:
1. Add Job
2. Delete Job
3. Display Jobs
4. Exit
Enter your choice: 4 Exiting...

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:12

TITLE: Write program to implement a priority queue in C++ using an inorder list to store the items in the queue. Create a class that includes the data items (which should be template) and the priority (which should be int). The inorder list should contain these objects, with operator <= overloaded so that the items with highest priority appear at the beginning of the list (which will make it relatively easy to retrieve the highest item.)

CODE:

```
#include <iostream>
#include <vector>
#include <algorithm>

template <typename T>
class Item { public:    T
data;
  int priority;

  Item(T data, int priority) : data(data), priority(priority) {}

  // Overloading the <= operator to compare priorities
bool operator<=(const Item<T>& other) const {
return this->priority <= other.priority;
  }
};

template <typename T> class
PriorityQueue { private:
  std::vector<Item<T>> items;

public:
  // Function to insert an item into the priority queue
  void enqueue(T data, int priority) {
Item<T> newItem(data, priority);
items.push_back(newItem);
    // Sort the items based on priority in descending order
    std::sort(items.begin(), items.end(), [](const Item<T>& a, const Item<T>& b) {
return b <= a; // Higher priority first
    });
  }

  // Function to remove and return the highest priority item
  T dequeue() {       if
(items.empty()) {
      throw std::runtime_error("Priority queue is empty");
    }
```

```
        T highestPriorityItem = items.front().data;
        items.erase(items.begin()); // Remove the item with the highest priority
return highestPriorityItem;
    }

    // Function to check if the priority queue is empty
bool isEmpty() const {
        return items.empty();
    }

    // Function to get the highest priority item without removing it
    T peek() const {
if (items.empty()) {
            throw std::runtime_error("Priority queue is empty");
        }
        return items.front().data;
    }

    // Function to display the items in the priority queue
    void display() const {        std::cout
<< "Priority Queue:\n";        for
(const auto& item : items) {
            std::cout << "Data: " << item.data << ", Priority: " << item.priority << "\n";
        }
    }
};

int main() {
    PriorityQueue<std::string> pq;

    pq.enqueue("Task 1", 2);
pq.enqueue("Task 2", 5);     pq.enqueue("Task
3", 1);     pq.enqueue("Task 4", 3);

    pq.display();

    std::cout << "\nDequeueing highest priority item: " << pq.dequeue() << "\n";
pq.display();

    std::cout << "\nPeek at highest priority item: " << pq.peek() << "\n";
return 0;
}
```

Output:

```
Priority Queue:
Data: Task 2, Priority: 5
Data: Task 4, Priority: 3
Data: Task 1, Priority: 2
Data: Task 3, Priority: 1
```

Dequeueing highest priority item: Task 2 Priority
Queue:
Data: Task 4, Priority: 3
Data: Task 1, Priority: 2
Data: Task 3, Priority: 1
Peek at highest priority item: Task 4

Name : Prem Chuniyan

Roll Number:75

ASSIGNMENT NO.:13

TITLE: 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>
#include <string>

using namespace std;

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

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

   ~CircularQueue() {
delete[] queue;
   }

   // Function to add an order
void enqueue(string order) {
if (size == capacity) {
         cout << "Order queue is full. Cannot accept new orders." << endl;
return;
      }
      queue[rear] = order;
rear = (rear + 1) % capacity;
      size++;
      cout << "Order added: " << order << endl;
   }

   // Function to serve an order
void dequeue() {       if (size ==
0) {
         cout << "No orders to serve." << endl;
return;
      }
      cout << "Serving order: " << queue[front] << endl;
```

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

    // Function to display current orders
void displayOrders() {        if (size ==
0) {
            cout << "No current orders." << endl;
            return;
        }
        cout << "Current orders in the queue: ";
        for (int i = 0; i < size; i++) {
            cout << queue[(front + i) % capacity] << " ";
        }
        cout << endl;
    }

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

int main() {
int M;
    cout << "Enter maximum number of orders (M): ";
cin >> M;

    CircularQueue orderQueue(M);
    int choice;
string order;

    do {
        cout << "\nMenu:\n";        cout << "1.
Place an order\n";        cout << "2. Serve
an order\n";        cout << "3. Display
current orders\n";
        cout << "4. Exit\n";        cout
<< "Enter your choice: ";
        cin >> choice;

        switch (choice) {
case 1:
                cout << "Enter the order: ";
                cin.ignore(); // To clear the newline character from input buffer
getline(cin, order);                orderQueue.enqueue(order);                break;
case 2:
                orderQueue.dequeue();
break;            case 3:
                orderQueue.displayOrders();
```

```
            break;
case 4:
            cout << "Exiting program." << endl;
            break;
default:
            cout << "Invalid choice! Please try again." << endl;
      }
   } while (choice != 4);

   return 0;
}
```

OUTPUT:

```
Enter maximum number of orders (M): 4
//Displaying Menu
//Placing order Menu:
1. Place an order
2. Serve an order
3. Display current orders 4. Exit
Enter your choice: 1
Enter the order: onion pizza
Order added: onion pizza
//Placing another order Menu:
1. Place an order
2. Serve an order
3. Display current orders
4. Exit
Enter your choice: 1
Enter the order: cheese pizza
Order added: cheese pizza
//Placing another order Menu:
1. Place an order
2. Serve an order
3. Display current orders 4. Exit
Enter your choice: 1
Enter the order: garlic bread
Order added: garlic bread
//Now serving the order Menu:
1. Place an order
2. Serve an order
3. Display current orders
4. Exit
Enter your choice: 2
Serving order: onion pizza
//Displaying the current orders Menu:
1. Place an order
2. Serve an order
3. Display current orders
4. Exit
```

Enter your choice: 3
Current orders in the queue: cheese pizza garlic bread
//Serving the order Menu:
1. Place an order
2. Serve an order
3. Display current orders 4. Exit
Enter your choice: 2
Serving order: cheese pizza
//Displaying the current orders Menu:
1. Place an order
2. Serve an order
3. Display current orders
4. Exit
Enter your choice: 3
Current orders in the queue: garlic bread
//Exiting the program Menu:
1. Place an order
2. Serve an order
3. Display current orders
4. Exit Enter your choice: 4 Exiting program.