Assignment 1

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: -

1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. Number of students who play neither cricket nor badminton
4. 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:

cricket = []

badminton = []

football = []

total\_cricket = int(input("Enter the number of students who play cricket-\n"))

for i in range(total\_cricket):

roll\_number\_cricket = int(input("Enter the roll number of the student-\n"))

if roll\_number\_cricket not in cricket:

cricket.append(roll\_number\_cricket)

else:

pass

total\_badminton = int(input("Enter the number of students who play badminton-\n"))

for i in range(total\_badminton):

roll\_number\_badminton = int(input("Enter the roll number of the student-\n"))

if roll\_number\_badminton not in badminton:

badminton.append(roll\_number\_badminton)

else:

pass

total\_football = int(input("Enter the number of students who play football-\n"))

for i in range(total\_football):

roll\_number\_football = int(input("Enter the roll number of the student-\n"))

if roll\_number\_football not in football:

football.append(roll\_number\_football)

else:

pass

def both\_cricket\_and\_badminton(cricket,badminton):

cricket\_and\_badminton = []

for i in cricket:

if i in badminton:

cricket\_and\_badminton.append(i)

for j in badminton:

if j in cricket and j not in cricket\_and\_badminton:

cricket\_and\_badminton.append(j)

return cricket\_and\_badminton

def cricket\_or\_badminton\_but\_not\_both(cricket,badminton):

cricket\_or\_badminton\_not\_both = []

for i in cricket:

if i not in badminton:

cricket\_or\_badminton\_not\_both.append(i)

else:

pass

for j in badminton:

if j not in cricket and j not in cricket\_or\_badminton\_not\_both:

cricket\_or\_badminton\_not\_both.append(j)

else:

pass

return cricket\_or\_badminton\_not\_both

def neither\_cricket\_nor\_badminton(cricket,badminton,football):

neither\_cricket\_nor\_badminton\_count = 0

for i in football:

if i not in cricket and i not in badminton:

neither\_cricket\_nor\_badminton\_count += 1

else:

pass

return neither\_cricket\_nor\_badminton\_count

def cricket\_and\_football\_not\_badminton(cricket,badminton,football):

cricket\_and\_football\_not\_badminton = []

for i in cricket:

if i in football and i not in badminton and i not in cricket\_and\_football\_not\_badminton:

cricket\_and\_football\_not\_badminton.append(i)

else:

pass

for j in football:

if j in cricket and j not in badminton and j not in cricket\_and\_football\_not\_badminton:

cricket\_and\_football\_not\_badminton.append(j)

cricket\_and\_football\_not\_badminton\_count = 0

for k in cricket\_and\_football\_not\_badminton:

cricket\_and\_football\_not\_badminton\_count += 1

return cricket\_and\_football\_not\_badminton\_count

print("Students who play both cricket and badminton- ",both\_cricket\_and\_badminton(cricket,badminton))

print("Students who play cricket or badminton but not both- ",cricket\_or\_badminton\_but\_not\_both(cricket,badminton))

print("Number of students who play neither cricket nor badminton- ",neither\_cricket\_nor\_badminton(cricket,badminton,football))

print("Number of students who play cricket and football but not badmininton- ",cricket\_and\_football\_not\_badminton(cricket,badminton,football))

OUTPUT

Enter the number of students who play cricket-

3

Enter the roll number of the student-

23

Enter the roll number of the student-

12

Enter the roll number of the student-

13

Enter the number of students who play badminton-

2

Enter the roll number of the student-

12

Enter the roll number of the student-

25

Enter the number of students who play football-

3

Enter the roll number of the student-

12

Enter the roll number of the student-

13

Enter the roll number of the student-

27

Students who play both cricket and badminton- [12]

Students who play cricket or badminton but not both- [23, 13, 25]

Number of students who play neither cricket nor badminton- 1

Number of students who play cricket and football but not badmininton- 1

Assignment 2

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:

1. The average score of class
2. Highest score and lowest score of class
3. Count of students who were absent for the test

Display mark with highest frequency

CODE:

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

marks = []

for i in range(n):

element = int(input("Enter the marks-\n"))

marks.append(element)

def average(marks):

sum = 0

len = 0

for i in marks:

if i != -1:

sum += i

for j in marks:

if j != -1:

len += 1

return sum/len

def maximum(marks):

max = -9999999

for i in marks:

if i == -1:

pass

elif i > max:

max = i

else:

pass

return max

def minimum(marks):

min = 9999999

for i in marks:

if i == -1:

pass

elif i < min:

min = i

else:

pass

return min

def max\_frequency(marks):

distinct\_elements = []

count\_of\_distinct\_elements = []

for i in marks:

if i not in distinct\_elements and i!=-1:

distinct\_elements.append(i)

else:

pass

for j in distinct\_elements:

count = 0

for k in marks:

if j == k:

count += 1

else:

pass

count\_of\_distinct\_elements.append(count)

max = -9999999

for l in count\_of\_distinct\_elements:

max\_index = 0

if l > max or l == max:

max = l

max\_index += 1

else:

pass

return distinct\_elements[max\_index],max

def absent\_students(marks):

count = 0

for i in marks:

if i == -1:

count += 1

else:

pass

return count

print("AVERAGE OF THE CLASS IS-\n",average(marks))

print("MAXIMUM MARKS ARE-\n",maximum(marks))

print("MINIMUM MARKS ARE-\n",minimum(marks))

print("COUNT OF STUDENTS WHO ARE ABSENT IS-\n",absent\_students(marks))

result = max\_frequency(marks)

print("MAXIMUM OCCURING MARKS AND ITS FREQUENCY IS-\n",result[0],result[1])

OUTPUT

Enter the number of students-

4

Enter the marks-

34

Enter the marks-

76

Enter the marks-

98

Enter the marks-

75

AVERAGE OF THE CLASS IS-

70.75

MAXIMUM MARKS ARE-

98

MINIMUM MARKS ARE-

34

COUNT OF STUDENTS WHO ARE ABSENT IS-

0

MAXIMUM OCCURING MARKS AND ITS FREQUENCY IS-

76 1

Assignment 3(matrix operation)

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

code:

# Function to add two matrices

def add\_matrices(A, B):

rows = len(A)

cols = len(A[0])

result = [[0 for \_ in range(cols)] for \_ in range(rows)]

for i in range(rows):

for j in range(cols):

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

return result

# Function to subtract two matrices

def subtract\_matrices(A, B):

rows = len(A)

cols = len(A[0])

result = [[0 for \_ in range(cols)] for \_ in range(rows)]

for i in range(rows):

for j in range(cols):

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

return result

# Function to multiply two matrices

def multiply\_matrices(A, B):

rows\_A = len(A)

cols\_A = len(A[0])

rows\_B = len(B)

cols\_B = len(B[0])

if cols\_A != rows\_B:

raise ValueError("Matrix dimensions must be compatible for multiplication.")

result = [[0 for \_ in range(cols\_B)] for \_ in range(rows\_A)]

for i in range(rows\_A):

for j in range(cols\_B):

for k in range(cols\_A):

result[i][j] += A[i][k] \* B[k][j]

return result

# Function to transpose a matrix

def transpose\_matrix(A):

rows = len(A)

cols = len(A[0])

result = [[0 for \_ in range(rows)] for \_ in range(cols)]

for i in range(rows):

for j in range(cols):

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

return result

# Example matrices for testing

A = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

B = [

[9, 8, 7],

[6, 5, 4],

[3, 2, 1]

]

# Perform operations

print("Matrix A:")

for row in A:

print(row)

print("\nMatrix B:")

for row in B:

print(row)

# a) Addition of matrices

print("\nAddition of A and B:")

add\_result = add\_matrices(A, B)

for row in add\_result:

print(row)

# b) Subtraction of matrices

print("\nSubtraction of A and B:")

subtract\_result = subtract\_matrices(A, B)

for row in subtract\_result:

print(row)

# c) Multiplication of matrices

print("\nMultiplication of A and B:")

multiply\_result = multiply\_matrices(A, B)

for row in multiply\_result:

print(row)

# d) Transpose of matrix A

print("\nTranspose of matrix A:")

transpose\_result = transpose\_matrix(A)

for row in transpose\_result:

print(row)

Matrix A:

[1, 2, 3]

[4, 5, 6]

[7, 8, 9]

Matrix B:

[9, 8, 7]

[6, 5, 4]

[3, 2, 1]

Addition of A and B:

[10, 10, 10]

[10, 10, 10]

[10, 10, 10]

Subtraction of A and B:

[-8, -6, -4]

[-2, 0, 2]

[4, 6, 8]

Multiplication of A and B:

[30, 24, 18]

[84, 69, 54]

[138, 114, 90]

Transpose of matrix A:

[1, 4, 7]

[2, 5, 8]

[3, 6, 9]

Assignment 4

1. Write a **Python** program to store roll numbers of student in array who attended training program in random order. Write function for searching whether particular student attended training program or not, using Linear search and Sentinel search.
2. Write a **Python** program to store roll numbers of student array who attended training

program in sorted order. Write function for searching whether particular student attended training program or not, using Binary search and Fibonacci search

code:

# Part (a) - Linear Search and Sentinel Search

def linear\_search(arr, target):

"""Linear search to find if target is in the array."""

for roll\_no in arr:

if roll\_no == target:

return True

return False

def sentinel\_search(arr, target):

"""Sentinel search to find if target is in the array."""

n = len(arr)

last = arr[-1]

arr[-1] = target # Set the sentinel to target value

i = 0

while arr[i] != target:

i += 1

arr[-1] = last # Restore the last element

# If i is less than n-1, we found the element

return i < n-1 or arr[-1] == target

# Example random roll numbers (not sorted)

students\_roll\_numbers = [102, 105, 107, 109, 104, 101, 103]

target\_roll\_number = 104

# Linear Search

print("Linear Search:")

if linear\_search(students\_roll\_numbers, target\_roll\_number):

print(f"Roll number {target\_roll\_number} attended the training program.")

else:

print(f"Roll number {target\_roll\_number} did NOT attend the training program.")

# Sentinel Search

print("\nSentinel Search:")

if sentinel\_search(students\_roll\_numbers, target\_roll\_number):

print(f"Roll number {target\_roll\_number} attended the training program.")

else:

print(f"Roll number {target\_roll\_number} did NOT attend the training program.")

# Part (b) - Binary Search and Fibonacci Search

def binary\_search(arr, target):

"""Binary Search to find if target is in the sorted array."""

low = 0

high = len(arr) - 1

while low <= high:

mid = low + (high - low) // 2

if arr[mid] == target:

return True

elif arr[mid] < target:

low = mid + 1

else:

high = mid - 1

return False

def fibonacci\_search(arr, target):

"""Fibonacci Search to find if target is in the sorted array."""

n = len(arr)

# Initialize fibonacci numbers

fib\_m\_minus\_2 = 0 # (m-2)th Fibonacci No.

fib\_m\_minus\_1 = 1 # (m-1)th Fibonacci No.

fib\_m = fib\_m\_minus\_1 + fib\_m\_minus\_2 # mth Fibonacci No.

# Find the largest Fibonacci number smaller than or equal to n

while fib\_m < n:

fib\_m\_minus\_2 = fib\_m\_minus\_1

fib\_m\_minus\_1 = fib\_m

fib\_m = fib\_m\_minus\_1 + fib\_m\_minus\_2

# Now fib\_m is the smallest Fibonacci number greater than or equal to n

offset = -1

while fib\_m > 1:

# Check if fib\_m\_minus\_2 is a valid index

i = min(offset + fib\_m\_minus\_2, n-1)

# If target is greater, move offset to i

if arr[i] < target:

fib\_m = fib\_m\_minus\_1

fib\_m\_minus\_1 = fib\_m\_minus\_2

fib\_m\_minus\_2 = fib\_m - fib\_m\_minus\_1

offset = i

# If target is smaller, move fib\_m to fib\_m\_minus\_2

elif arr[i] > target:

fib\_m = fib\_m\_minus\_2

fib\_m\_minus\_1 -= fib\_m\_minus\_2

fib\_m\_minus\_2 = fib\_m - fib\_m\_minus\_1

else:

return True

# Check if the last element is the target

if fib\_m\_minus\_1 and arr[offset + 1] == target:

return True

return False

# Example sorted roll numbers (ascending order)

students\_roll\_numbers\_sorted = [101, 102, 103, 104, 105, 107, 109]

target\_roll\_number = 104

# Binary Search

print("\nBinary Search:")

if binary\_search(students\_roll\_numbers\_sorted, target\_roll\_number):

print(f"Roll number {target\_roll\_number} attended the training program.")

else:

print(f"Roll number {target\_roll\_number} did NOT attend the training program.")

# Fibonacci Search

print("\nFibonacci Search:")

if fibonacci\_search(students\_roll\_numbers\_sorted, target\_roll\_number):

print(f"Roll number {target\_roll\_number} attended the training program.")

else:

print(f"Roll number {target\_roll\_number} did NOT attend the training program.")

output:

Linear Search:

Roll number 104 attended the training program.

Sentinel Search:

Roll number 104 attended the training program.

Binary Search:

Roll number 104 attended the training program.

Fibonacci Search:

Roll number 104 attended the training program.

Assignment 5

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

1. Selection Sort

b) Bubble sort and display top five scores.

CODE:

def selection\_sort(percentage):

print("Original array is ", percentage)

for i in range(len(percentage)-1):

minimum = i

for j in range(i+1, len(percentage), 1):

if (percentage[j] < percentage[minimum]):

minimum = j

percentage[i],percentage[minimum] = percentage[minimum],percentage[i]

print("Pass " + str(i+1) + ":" , percentage)

return percentage

def bubble\_sort(percentage):

print("Original array is ", percentage)

for i in range(len(percentage)-1):

for j in range(len(percentage)-i-1):

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

percentage[j],percentage[j+1] = percentage[j+1],percentage[j]

print("Pass " + str(i+1) + ":", percentage)

return percentage

n = int(input("Enter the number of students here-\n"))

percentage = []

for i in range(n):

element = float(input("Enter the percentage of the student-\n"))

if element > 0:

percentage.append(element)

choice = int(input("Enter 0 if you want to use selection sort and enter 1 if you want to use bubble sort-\n"))

if choice == 0:

selection\_sort(percentage)

else:

bubble\_sort(percentage)

print("The percentages of the students after sorting are-", percentage)

print("The percentages of top 5 students are-", percentage[-1:-6:-1])

OUTPUT

Enter the number of students here-

5

Enter the percentage of the student-

34.7

Enter the percentage of the student-

56

Enter the percentage of the student-

78.9

Enter the percentage of the student-

98.2

Enter the percentage of the student-

56.7

Enter 0 if you want to use selection sort and enter 1 if you want to use bubble sort-

0

Original array is [34.7, 56.0, 78.9, 98.2, 56.7]

Pass 1: [34.7, 56.0, 78.9, 98.2, 56.7]

Pass 2: [34.7, 56.0, 78.9, 98.2, 56.7]

Pass 3: [34.7, 56.0, 56.7, 98.2, 78.9]

Pass 4: [34.7, 56.0, 56.7, 78.9, 98.2]

The percentages of the students after sorting are- [34.7, 56.0, 56.7, 78.9, 98.2]

The percentages of top 5 students are- [98.2, 78.9, 56.7, 56.0, 34.7]

Enter the number of students here-

3

Enter the percentage of the student-

56

Enter the percentage of the student-

67.8

Enter the percentage of the student-

78.5

Enter 0 if you want to use selection sort and enter 1 if you want to use bubble sort-

1

Original array is [56.0, 67.8, 78.5]

Pass 1: [56.0, 67.8, 78.5]

Pass 2: [56.0, 67.8, 78.5]

The percentages of the students after sorting are- [56.0, 67.8, 78.5]

The percentages of top 5 students are- [78.5, 67.8, 56.0]

Assignment 6

Write **Python** program to store 10th class percentage of students in array. Write function for sorting array of floating point numbers in ascending order using radix sort and display top five scores

# Function to perform counting sort for a specific digit place

def counting\_sort(arr, exp):

n = len(arr)

output = [0] \* n # Output array to store sorted numbers

count = [0] \* 10 # Count array for digits (0-9)

# Count the occurrences of digits

for i in range(n):

index = (arr[i] // exp) % 10

count[index] += 1

# Change count[i] to contain the actual position of this digit in output[]

for i in range(1, 10):

count[i] += count[i - 1]

# Build the output array

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

index = (arr[i] // exp) % 10

output[count[index] - 1] = arr[i]

count[index] -= 1

# Copy the sorted numbers into the original array

for i in range(n):

arr[i] = output[i]

# Function to perform Radix Sort

def radix\_sort(arr):

# Find the maximum number to determine the number of digits

max\_num = max(arr)

# Apply counting sort for every digit

exp = 1

while max\_num // exp > 0:

counting\_sort(arr, exp)

exp \*= 10

# Function to display the top 5 scores

def display\_top\_five(arr):

print("\nTop 5 scores:")

for i in range(min(5, len(arr))):

print(f"Rank {i + 1}: {arr[i]}%")

# Storing 10th class percentages in an array

student\_percentages = [88.5, 92.3, 75.0, 85.4, 90.1, 78.8, 95.6, 80.2, 88.7, 91.3]

# Print original list

print("Original List of Percentages:")

print(student\_percentages)

# Convert floating-point percentages to integers by multiplying by 100 to avoid float sorting issues

scaled\_percentages = [int(x \* 100) for x in student\_percentages]

# Sort using Radix Sort

radix\_sort(scaled\_percentages)

# Convert back to float percentages

sorted\_percentages = [x / 100 for x in scaled\_percentages]

# Print the sorted list

print("\nSorted List of Percentages:")

print(sorted\_percentages)

# Display the top 5 scores

display\_top\_five(sorted\_percentages)

output: Original List of Percentages:

[88.5, 92.3, 75.0, 85.4, 90.1, 78.8, 95.6, 80.2, 88.7, 91.3]

Sorted List of Percentages:

[75.0, 78.8, 80.2, 85.4, 88.5, 88.7, 90.1, 91.3, 92.3, 95.6]

Top 5 scores:

Rank 1: 75.0%

Rank 2: 78.8%

Rank 3: 80.2%

Rank 4: 85.4%

Rank 5: 88.5%

ASSIGNMENT 7( PINACALE CLUB)

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:

1. Add and delete the members as well as president or even secretary.
2. Compute total number of members of club
3. Display members
4. Two linked lists exists for two divisions. Concatenate two lists.

CODE:

#include<bits/stdc++.h>

using namespace std;

struct Node

{

pair<string,int> data;

Node \*next;

};

void insertMember(Node \*president, Node \*member)

{

Node \*temp = president;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = member;

}

void createMember(Node \*president)

{

string name;

int prn;

cout << "Enter the name-\n";

cin >> name;

cout << "Enter the prn-\n";

cin >> prn;

Node \*member = new Node;

member->data.first = name;

member->data.second = prn;

member->next = NULL;

insertMember(president, member);

}

void display(Node \*president)

{

Node \*temp1 = president;

while (temp1 != NULL)

{

cout << temp1->data.first << " " << temp1->data.second << endl;

temp1 = temp1->next;

}

}

void totalMembers(Node \*president)

{

int count = 0;

Node \*temp = president;

while (temp != NULL)

{

count++;

temp = temp->next;

}

cout << "The total members in the club are- " << count << endl;

}

void deleteMember(Node \*&president)

{

Node \*temp = president;

int position;

cout << "Enter the position of the member to be deleted-\n";

cin >> position;

int count = 0;

if (president == NULL)

{

cout << "The member linked list is already empty-\n";

}

else if (position == 0)

{

president = temp->next;

delete temp;

}

else

{

while (count != position-1)

{

temp = temp->next;

count++;

}

if (temp->next == NULL)

{

cout << "Element is not present in the member linked list" << endl;

}

Node \*todelete = temp->next;

temp->next = temp->next->next;

delete todelete;

}

}

Node\* concatenate(Node \*president1, Node \*president2)

{

Node \*head = president1;

while (head->next != NULL)

{

head = head->next;

}

head->next = president2;

return president1;

}

int main()

{

Node \*president1 = new Node();

Node \*president2 = new Node();

string president1\_name;

int president1\_prn;

cout << "Enter president-1's name-\n";

cin >> president1\_name;

cout << "Enter president-1's prn-\n";

cin >> president1\_prn;

president1->data.first = president1\_name;

president1->data.second = president1\_prn;

president1->next = NULL;

int members1;

cout << "Enter number of members except president-1-\n";

cin >> members1;

while (members1 != 0)

{

createMember(president1);

members1--;

}

string president2\_name;

int president2\_prn;

cout << "Enter president-2's name-\n";

cin >> president2\_name;

cout << "Enter president-2's prn-\n";

cin >> president2\_prn;

president2->data.first = president2\_name;

president2->data.second = president2\_prn;

president2->next = NULL;

int members2;

cout << "Enter number of members except president-2-\n";

cin >> members2;

while (members2 != 0)

{

createMember(president2);

members2--;

}

char decider;

cout << "Enter Y for starting/continuing or N for exitting-\n";

cin >> decider;

while (decider == 'Y')

{

int choice;

int president\_choice;

cout << "1- Display the members\n2- Display total members\n3- Delete a member\n4- Concatenate the two member lists\n";

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter 1 for president 1 and 2 for president 2-\n";

cin >> president\_choice;

if (president\_choice == 1)

{

display(president1);

break;

}

else

{

display(president2);

break;

}

case 2:

cout << "Enter 1 for president 1 and 2 for president 2-\n";

cin >> president\_choice;

if (president\_choice == 1)

{

totalMembers(president1);

break;

}

else

{

totalMembers(president2);

break;

}

case 3:

cout << "Enter 1 for president 1 and 2 for president 2-\n";

cin >> president\_choice;

if (president\_choice == 1)

{

deleteMember(president1);

break;

}

else

{

deleteMember(president2);

break;

}

case 4:

cout << "Concatenating the two member linked lists....\n";

concatenate(president1,president2);

break;

default:

break;

}

cout << "Enter Y for starting/continuing or N for exitting-\n";

cin >> decider;

}

cout << "The program has exitted!";

return 0;

}

# ASSIGNMENT 8( SETOPERATION-VANILLA)

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-

1. Set of students who like both vanilla and butterscotch
2. Set of students who like either vanilla or butterscotch or not both

Number of students who like neither vanilla nor butterscotch

CODE:

#include <bits/stdc++.h>

using namespace std;

class node

{

public:

int rollno;

node \*next;

node(int num) //constructor

{

rollno = num;

next = NULL;

}

}\*all\_head = NULL, \*vanilla\_head = NULL, \*butter\_head = NULL, \*union\_head = NULL, \*inters\_head = NULL, \*diff\_head = NULL;

void append(node \*\*head\_ref) //insert at end

{

int num;

cout<<"Enter Rollno: ";

cin>>num;

node \*new\_node = new node(num);

if (\*head\_ref == NULL) //if list is empty

{

\*head\_ref = new\_node;

return;

}

node \*last = \*head\_ref;

while (last -> next != NULL)

{

last = last -> next;

}

last -> next = new\_node;

}

void append(node \*\*head\_ref, int key)

{

node \*new\_node = new node(key);

if (\*head\_ref == NULL) //if list is empty

{

\*head\_ref = new\_node;

return;

}

node \*last = \*head\_ref;

while (last -> next != NULL)

{

last = last -> next;

}

last -> next = new\_node;

}

void display\_list(node \*\*head\_ref)

{

cout<<"\n";

node \*temp = \*head\_ref;

if (temp == NULL)

{

cout<<"List is empty";

return;

}

while (temp != NULL)

{

cout<<temp -> rollno<<" ";

temp = temp -> next;

}

cout<<"\n";

}

void get\_all()

{

int count;

cout<<"\nEnter total students: ";

cin>>count;

for (int i = 0; i < count; i++)

{

append(&all\_head);

}

cout<<"\nAll students: ";

display\_list(&all\_head);

}

void get\_vanilla()

{

int count;

cout<<"\nEnter total students who like vanilla: ";

cin>>count;

for (int i = 0; i < count; i++)

{

append(&vanilla\_head);

}

cout<<"\nStudents who like vanilla: ";

display\_list(&vanilla\_head);

}

void get\_butter()

{

int count;

cout<<"\nEnter total students who like butterscotch: ";

cin>>count;

for (int i = 0; i < count; i++)

{

append(&butter\_head);

}

cout<<"\nStudents who like butterscotch: ";

display\_list(&butter\_head);

}

node \*\*unio(node \*\*head1, node \*\*head2)

{

//fill union list with all elements of list 1

node \*temp1 = \*head1;

while (temp1 != NULL)

{

append(&union\_head, (temp1 -> rollno));

temp1 = temp1 -> next;

}

//fill union list with non-common elements from list 2

node \*temp2 = \*head2;

while (temp2 != NULL)

{

temp1 = \*head1;

bool flag = false;

while (temp1 != NULL)

{

if ((temp2 -> rollno) == (temp1 -> rollno))

flag = true;

temp1 = temp1 -> next;

}

if (!flag)

{

append(&union\_head, (temp2 -> rollno));

}

temp2 = temp2 -> next;

}

return &union\_head;

}

node \*\*inters(node \*\*head1, node \*\*head2)

{

node \*temp1 = \*head1;

while (temp1 != NULL)

{

node \*temp2 = \*head2;

while (temp2 != NULL)

{

if ((temp1 -> rollno) == (temp2 -> rollno))

{

append(&inters\_head, (temp1 -> rollno));

}

temp2 = temp2 -> next;

}

temp1 = temp1 -> next;

}

return &inters\_head;

}

node \*\*diff(node \*\*head1, node \*\*head2)

{

node \*temp1 = \*head1;

while (temp1 != NULL)

{

node \*temp2 = \*head2;

bool flag = false;

while (temp2 != NULL)

{

if ((temp1 -> rollno) == (temp2 -> rollno))

{

flag = true;

}

temp2 = temp2 -> next;

}

if(!flag)

{

append(&diff\_head, (temp1 -> rollno));

}

temp1 = temp1 -> next;

}

return &diff\_head;

}

int main()

{

bool exit = false;

int choice;

do

{

cout<<"\n1. To enter data";

cout<<"\n2. To display the rollnos of students who like vanila or butterscotch";

cout<<"\n3. To display the rollnos of students who like only vanila";

cout<<"\n4. To display the rollnos of students who like only butterscotch";

cout<<"\n5. To display the rollnos of students who like both vanila and butterscotch ";

cout<<"\n6. To display the rollnos of students who neither like vanila nor butterscotch";

cout<<"\n7. Exit";

cout<<"\nEnter choice: ";

cin>>choice;

switch (choice)

{

case 1:

get\_all();

get\_vanilla();

get\_butter();

break;

case 2:

cout<<"Students who like vanilla or butterscotch: ";

display\_list( unio(&vanilla\_head, &butter\_head) );

break;

case 3:

cout<<"Students who like only vanilla: ";

display\_list(&vanilla\_head);

break;

case 4:

cout<<"Students who like butterscotch: ";

display\_list(&butter\_head);

break;

case 5:

cout<<"Students who like both vanilla and butterscotch: ";

display\_list( inters(&vanilla\_head, &butter\_head) );

break;

case 6:

cout<<"Students who like neither vanilla nor butterscotch: ";

display\_list( diff(&all\_head, unio(&vanilla\_head, &butter\_head)) );

break;

case 7:

exit = true;

break;

}

} while (!exit);

}

OUTPUT

. To enter data

2. To display the rollnos of students who like vanila or butterscotch

3. To display the rollnos of students who like only vanila

4. To display the rollnos of students who like only butterscotch

5. To display the rollnos of students who like both vanila and butterscotch

6. To display the rollnos of students who neither like vanila nor butterscotch

7. Exit

Enter choice: 1

Enter total students: 2

Enter Rollno: 12

Enter Rollno: 23

All students:

12 23

Enter total students who like vanilla:

1

Enter Rollno: 12

Students who like vanilla:

12

Enter total students who like butterscotch:

0

Students who like butterscotch:

List is empty

1. To enter data

2. To display the rollnos of students who like vanila or butterscotch

3. To display the rollnos of students who like only vanila

4. To display the rollnos of students who like only butterscotch

5. To display the rollnos of students who like both vanila and butterscotch

6. To display the rollnos of students who neither like vanila nor butterscotch

7. Exit

Enter choice: 2

Students who like vanilla or butterscotch:

12

1. To enter data

2. To display the rollnos of students who like vanila or butterscotch

3. To display the rollnos of students who like only vanila

4. To display the rollnos of students who like only butterscotch

5. To display the rollnos of students who like both vanila and butterscotch

6. To display the rollnos of students who neither like vanila nor butterscotch

7. Exit

Enter choice: 4

Students who like butterscotch:

List is empty

1. To enter data

2. To display the rollnos of students who like vanila or butterscotch

3. To display the rollnos of students who like only vanila

4. To display the rollnos of students who like only butterscotch

5. To display the rollnos of students who like both vanila and butterscotch

6. To display the rollnos of students who neither like vanila nor butterscotch

7. Exit

Enter choice: 6

Students who like neither vanilla nor butterscotch:

23

1. To enter data

2. To display the rollnos of students who like vanila or butterscotch

3. To display the rollnos of students who like only vanila

4. To display the rollnos of students who like only butterscotch

5. To display the rollnos of students who like both vanila and butterscotch

6. To display the rollnos of students who neither like vanila nor butterscotch

7. Exit

Enter choice: 7

ASSIGNMENT 9 (Stack parenthesis)

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>

using namespace std;

const int MAX=20;

class Stack

{

char str[MAX];

int top;

public:

Stack()

{

top=-1;

}

void push(char ch);

char pop();

char getTop();

bool isEmpty();

bool isFull();

void display();

void checkParenthesis();

};

bool Stack::isEmpty()

{

if(top==-1)

return 1;

else return 0;

}

bool Stack::isFull()

{

if(top==MAX-1)

return 1;

else

return 0;

}

void Stack :: display()

{

if(isEmpty()==1)

cout<<"\nStack is empty";

else

{

for(int i=0;i<=top;i++)

{

cout<<" "<<str[i];

}

}

}

void Stack::push(char ch)

{

if(!isFull())

{

top++;

str[top]=ch;

}

}

char Stack::pop()

{

if(!isEmpty())

{

char ch=str[top];

top--;

return ch;

}

else

{

return '\0';

}

}

void Stack::checkParenthesis()

{

cout<<"\nEnter # as a deliminator after expression(At the end)\n";

cout<<"\nEnter Expression: ";

cin.getline(str,MAX,'#');

char ch;

bool flag=0;

for(int i=0;str[i]!='\0';i++)

{

if(str[i]=='(' || str[i]=='[' || str[i]=='{')

push(str[i]);

if(str[i]==')'||str[i]==']'||str[i]=='}')

{

ch=pop();

if((str[i]==')'&& ch!='(') ||(str[i]==']'&& ch!='[')||(str[i]=='}'&& ch!='{'))

{

cout<<"\nNot parenthesized At "<<i<<" = "<<str[i];

flag=1;

break;

}

}

}

if(isEmpty()==1 && flag==0)

cout<<"\nExpresseion is Well Parenthesized.";

else

cout<<"\nExpression is not Well Parenthesized.";

}

int main()

{

int choice;

do

{

Stack s;

s.checkParenthesis();

cout<<"\nDO you want to continue?{1/0)";

cin>>choice;

}while(choice!=0);

return 0;

}

OUTPUT

Enter # as a deliminator after expression(At the end)

Enter Expression: (a+[C+P])#

Expresseion is Well Parenthesized.

DO you want to continue?{1/0)1

Enter # as a deliminator after expression(At the end)

Enter Expression: A{[c+d]#

Expression is not Well Parenthesized.

DO you want to continue?{1/0)

ASSIGNMENT10 (INFIX TO POSTFIX)

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.

Only '+', '-', '\*' and '/ ' operators are expected.

CODE:

#include<iostream>

#include<stdlib.h>

#include<string.h>

using namespace std;

class stack

{

private :

char c[100];

int max,top;

public :

stack(int m)

{

if (m > 1)

max = m;

else

{

cout<<"Capacity should be a positive natural number.";

return;

}

top = -1;

}

void push(char a)

{

if (top == -1)

{

c[0] = a;

top = 0;

}

else

{

if(top < max-1)

{

top = top + 1;

c[top] = a;

}

//else

// cout<<"Error : Capacity full!"<<endl;

}

}

char pop ()

{

if (top == -1)

{

//cout<<"Error : The stack is empty.";

char b = '#';

return b;

}

else

{

if (top < max)

{

char t = c[top];

c[top] = '\0';

top = top - 1;

return t;

}

/\*if (top == 0 || top == max+1)

{

cout<<"Error : The stack is empty."<<endl;

}\*/

}

}

char topelement ()

{

if (top == -1)

{

//cout<<"Error : The stack is empty.";

char b = '#';

return b;

}

char t = c[top];

return t;

}

int isempty()

{

if(top == -1)

{

return 1;

}

else

return 0;

}

int isfull()

{

if(top == max-1)

{

return 1;

}

else

return 0;

}

void display()

{

int a = top;

if (a == -1)

{

cout<<"Error : The stack is empty."<<endl;

return;

}

while (a != -1)

{

cout<<c[a]<<"\n";

a = a - 1;

}

}

};

class expression

{

char expr[50];

public :

expression()

{

expr[0] = '\0';

}

expression(char \*s)

{

strcpy(expr,s);

}

expression(string s)

{

int i = s.length();

for(int j=0; j<i; j++)

expr[j] = s.at(j);

expr[i] = '\0';

}

int FindPriority (char a)

{

switch(a)

{

case '/' :

return 4;

case '\*' :

return 4;

case '+' :

return 1;

case '-' :

return 1;

case '(' :

return -1;

case ')' :

return -2;

default :

return 0;

}

}

int Verify ()

{

int i,count1 = 0,count2 = 0;

for (i=0; expr[i]; i++)

{

if (expr[i] == '(')

count1++;

if (expr[i] == ')')

count2++;

}

if(count1 == count2)

return 1;

else

return 0;

}

string Infix2Postfix()

{

stack ob(50);

int i,pri,k = 0;

char postfix[50];

for (i = 0; expr[i] != '\0'; i++)

{

pri = FindPriority(expr[i]);

if (expr[i] == '(')

{

ob.push(expr[i]);

}

if (expr[i] == ')')

{

while(ob.isempty() != 1 && ob.topelement() != '(')

{

postfix[k] = ob.topelement();

ob.pop();

k++;

}

if (ob.isempty() != 1)

{

ob.pop();

}

}

if (pri == 0)

{

postfix[k] = expr[i];

k++;

}

if (pri == 1 || pri == 2 || pri == 3 || pri == 4)

{

if (ob.isempty() == 1)

{

ob.push(expr[i]);

}

else

{

while (ob.isempty() != 1 && ob.topelement() != '(' && pri <= FindPriority(ob.topelement()))

{

postfix[k] = ob.topelement();

ob.pop();

k++;

}

ob.push(expr[i]);

}

}

}

while (ob.isempty() != 1 )

{

postfix[k] = ob.topelement();

ob.pop();

k++;

}

postfix[k] = '\0';

return postfix;

}

float evaluation (string p)

{

int n;

cout<<"Enter the number of variables in the expression : ";

cin>>n;

cout<<"Enter the value of variables in the format : variable value (Eg. a 5 b 4 c 3)"<<endl;

char variable[10];

int i,j,k,pri;

float value [10];

for (i = 0; i < n; i++)

{

cin>>variable[i]>>value[i];

}

expression post(p);

stack obj(50);

int pos1,pos2;

float num1,num2;

float res;

char r = 'q';

for (i = 0; post.expr[i] ; i++)

{

pri = FindPriority(post.expr[i]);

cout<<post.expr[i]<<"\t"<<pri<<endl;

if (pri == 0)

{

obj.push(post.expr[i]);

cout<<"push operand "<<post.expr[i]<<endl;

}

else// (pri == 1 || pri == 2 || pri == 3 || pri == 4)

{

char x = obj.pop();

char y = obj.pop();

cout<<"popped 2 operands "<<x<<"\t"<<y<<endl;

for(j = 0; variable[j]; j++)

{

if (variable[j] == x)

{

pos1 = j;

break;

}

}

for(k = 0; variable[k]; k++)

{

if (variable[k] == y)

{

pos2 = k;

break;

}

}

num1 = value[pos1];

num2 = value[pos2];

switch(post.expr[i])

{

case '/' :

res = num2/num1;

cout<<"result "<<res<<endl;

break;

case '\*' :

res = num2\*num1;cout<<"result "<<res<<endl;

break;

case '+' :

res = num2+num1;cout<<"result "<<res<<endl;

break;

case '-' :

res = num2-num1;cout<<"result "<<res<<endl;

break;

}

//cout<<n;

int a = int(r);

a++;

r = char(a);

variable[n] = r;

value[n] = res;

//if(obj.isempty() == 1)

obj.push(variable[n]);

cout<<"push "<<variable[n]<<endl;

n++;

//int(r) = r + 1;

}

cout<<"display"<<endl;

obj.display();

cout<<endl<<endl;

}

for(j = 0; variable[j]; j++)

{

//cout<<"variable"<<variable[j]<<endl;

if (variable[j] == obj.topelement())

{

pos1 = j;

break;

}

}

return value[pos1];

}

};

int main ()

{

int choice;

char e[50];

do{

cout<<"Enter the expression : ";

cin>>e;

expression obj(e);

int t = obj.Verify();

if (t == 0)

{

cout<<"Error : Unbalanced parenthesis in the expression."<<endl;

return 0;

}

string p = obj.Infix2Postfix();

cout<<"The infix notation is : "<<e<<endl;

cout<<"The postfix notation is : "<<p<<endl;

float result = obj.evaluation(p);

cout<<"The result is : "<<result<<endl<<endl;

cout<<"Press 1 to continue \t Press 0 to exit \t";

cin>>choice;

}while(choice == 1);

return 0;

}

OUTPUT

Enter the expression : (A+C-D)\*(D-[A+C])

The infix notation is : (A+C-D)\*(D-[A+C])

The postfix notation is : AC+D-D[A-C]+\*

ASSIGNMENT 11( JOB QUEUE)

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>

#define MAX 10

using namespace std;

struct queue

{

int data[MAX];

int front,rear;

};

class Queue

{ struct queue q;

public:

Queue(){q.front=q.rear=-1;}

int isempty();

int isfull();

void enqueue(int x );

int delqueue();

void display();

};

int Queue::isempty()

{

return(q.front==q.rear)?1:0;

}

int Queue::isfull()

{ return(q.rear==MAX-1)?1:0;}

void Queue::enqueue(int x)

{q.data[++q.rear]=x;}

int Queue::delqueue()

{return q.data[++q.front];}

void Queue::display()

{ int i;

cout<<"\n";

for(i=q.front+1;i<=q.rear;i++)

cout<<q.data[i]<<" ";

}

int main()

{ Queue obj;

int ch,x;

do{ cout<<"\n 1.insert job\n 2.delete job\n 3.display\n 4.Exit\n Enter your choice:";

cin>>ch;

switch(ch)

{ case 1: if (!obj.isfull())

{ cout<<"\n Enter data:";

cin>>x;

obj.enqueue(x);

}

else

cout<< "Queue is overflow";

break;

case 2: if(!obj.isempty())

cout<<"\n Deleted Element="<<obj.delqueue();

else

{ cout<<"\n Queue is underflow"; }

cout<<"\nremaining jobs :";

obj.display();

break;

case 3: if (!obj.isempty())

{ cout<<"\n Queue contains:";

obj.display();

}

else

cout<<"\n Queue is empty";

break;

case 4: cout<<"\n Exit";

}

}while(ch!=4);

return 0;

}

OUTPUT

1.insert job

2.delete job

3.display

4.Exit

Enter your choice:3

Queue is empty

1.insert job

2.delete job

3.display

4.Exit

Enter your choice:1

Enter data:123

1.insert job

2.delete job

3.display

4.Exit

Enter your choice:3

Queue contains:

123

1.insert job

2.delete job

3.display

4.Exit

Enter your choice:1

Enter data:

12

1.insert job

2.delete job

3.display

4.Exit

Enter your choice:3

Queue contains:

123 12

1.insert job

2.delete job

3.display

4.Exit

Enter your choice:2

Deleted Element=123

remaining jobs :

12

1.insert job

2.delete job

3.display

4.Exit

Enter your choice

ASSIGNMENT 12( DEQUEUE)

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>

#include<stdio.h>

#define MAX 10

using namespace std;

struct que

{

int arr[MAX];

int front,rear;

};

void init(struct que \*q)

{

q->front=-1;

q->rear=-1;

}

void print(struct que q)

{

int i;

i=q.front;

while(i!=q.rear)

{

cout<<"\t"<<q.arr[i];

i=(i+1)%MAX;

}

cout<<"\t"<<q.arr[q.rear];

}

int isempty(struct que q)

{

return q.rear==-1?1:0;

}

int isfull(struct que q)

{

return (q.rear+1)%MAX==q.front?1:0;

}

void addf(struct que \*q,int data)

{

if(isempty(\*q))

{

q->front=q->rear=0;

q->arr[q->front]=data;

}

else

{

q->front=(q->front-1+MAX)%MAX;

q->arr[q->front]=data;

}

}

void addr(struct que \*q,int data)

{

if(isempty(\*q))

{

q->front=q->rear=0;

q->arr[q->rear]=data;

}

else

{

q->rear=(q->rear+1)%MAX;

q->arr[q->rear]=data;

}

}

int delf(struct que \*q)

{

int data1;

data1=q->arr[q->front];

if(q->front==q->rear)

init(q);

else

q->front=(q->front+1)%MAX;

return data1;

}

int delr(struct que \*q)

{

int data1;

data1=q->arr[q->rear];

if(q->front==q->rear)

init(q);

else

q->rear=(q->rear-1+MAX)%MAX;

return data1;

}

int main()

{

struct que q;

int data,ch;

init(&q);

while(ch!=6)

{

cout<<"\t\n1.Insert front"

"\t\n2.Insert rear"

"\t\n3.Delete front"

"\t\n4.Delete rear"

"\t\n5.Print"

"\t\n6.Exit";

cout<<"\nEnter your choice : ";

cin>>ch;

switch(ch)

{

case 1:

cout<<"\nEnter data to insert front : ";

cin>>data;

addf(&q,data);

break;

case 2:

cout<<"\nEnter the data to insert rear : ";

cin>>data;

addr(&q,data);

break;

case 3:

if(isempty(q))

cout<<"\nDequeue is empty!!!";

else

{

data=delf(&q);

cout<<"\nDeleted data is : "<<data;

}

break;

case 4:

if(isempty(q))

cout<<"\nDequeue is empty!!!";

else

{

data=delr(&q);

cout<<"\nDeleted data is : "<<data;

}

break;

case 5:

if(isempty(q))

cout<<"\nDequeue is empty!!!";

else

{

cout<<"\nDequeue elements are : ";

print(q);

}

break;

}

}

return 0;

}

OUTPUT

#include<iostream>

#include<stdio.h>

#define MAX 10

using namespace std;

struct que

{

int arr[MAX];

int front,rear;

};

void init(struct que \*q)

{

q->front=-1;

q->rear=-1;

}

void print(struct que q)

{

int i;

i=q.front;

while(i!=q.rear)

{

cout<<"\t"<<q.arr[i];

i=(i+1)%MAX;

}

cout<<"\t"<<q.arr[q.rear];

}

int isempty(struct que q)

{

return q.rear==-1?1:0;

}

int isfull(struct que q)

{

return (q.rear+1)%MAX==q.front?1:0;

}

void addf(struct que \*q,int data)

{

if(isempty(\*q))

{

q->front=q->rear=0;

q->arr[q->front]=data;

}

else

{

q->front=(q->front-1+MAX)%MAX;

q->arr[q->front]=data;

}

}

void addr(struct que \*q,int data)

{

if(isempty(\*q))

{

q->front=q->rear=0;

q->arr[q->rear]=data;

}

else

{

q->rear=(q->rear+1)%MAX;

q->arr[q->rear]=data;

}

}

int delf(struct que \*q)

{

int data1;

data1=q->arr[q->front];

if(q->front==q->rear)

init(q);

else

q->front=(q->front+1)%MAX;

return data1;

}

int delr(struct que \*q)

{

int data1;

data1=q->arr[q->rear];

if(q->front==q->rear)

init(q);

else

q->rear=(q->rear-1+MAX)%MAX;

return data1;

}

int main()

{

struct que q;

int data,ch;

init(&q);

while(ch!=6)

{

cout<<"\t\n1.Insert front"

"\t\n2.Insert rear"

"\t\n3.Delete front"

"\t\n4.Delete rear"

"\t\n5.Print"

"\t\n6.Exit";

cout<<"\nEnter your choice : ";

cin>>ch;

switch(ch)

{

case 1:

cout<<"\nEnter data to insert front : ";

cin>>data;

addf(&q,data);

break;

case 2:

cout<<"\nEnter the data to insert rear : ";

cin>>data;

addr(&q,data);

break;

case 3:

if(isempty(q))

cout<<"\nDequeue is empty!!!";

else

{

data=delf(&q);

cout<<"\nDeleted data is : "<<data;

}

break;

case 4:

if(isempty(q))

cout<<"\nDequeue is empty!!!";

else

{

data=delr(&q);

cout<<"\nDeleted data is : "<<data;

}

break;

case 5:

if(isempty(q))

cout<<"\nDequeue is empty!!!";

else

{

cout<<"\nDequeue elements are : ";

print(q);

}

break;

}

}

return 0;

}

ASSIGNMENT13( PIZZA PARLOR-CIRCULAR QUEUE)

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<bits/stdc++.h>

using namespace std;

class pizza

{

int front,rear,q[5];

public:

pizza()

{

front=-1;

rear=-1;

}

int isfull()

{

if((front==0&&rear==4)||front==rear+1)

{

return 1;

}

else

{

return 0;

}

}

int isempty()

{

if(front==-1&&rear==-1)

{

return 1;

}

else

{

return 0;

}

}

void add()

{

if(isfull()==0)

{

cout<<"\n Enter the Pizza ID: ";

if(front==-1&&rear==-1)

{

front=0;

rear=0;

cin>>q[rear];

}

else

{

rear=(rear+1)%5;

cin>>q[rear];

}

char c;

cout<<" Do you want to add another order ? ";

cin>>c;

if(c=='y'||c=='Y')

add();

}

else

{

cout<<"\n Orders are full ";

}

}

void serve()

{

if(isempty()==0)

{

if(front==rear)

{

cout<<"\n Order served is : "<<q[front];

front=-1;

rear=-1;

}

else

{

cout<<"\n Order served is : "<<q[front];

front=(front+1)%5;

}

}

else

{

cout<<"\n Orders are empty ";

}

}

void display()

{

if(isempty()==0)

{

for(int

i=front;i!=rear;i=(i+1)%5)

{

cout<<q[i]<<"<- ";

}

cout<<q[rear];

}

else

{

cout<<"\n Orders are empty";

}

}

void check()

{

int ch;

cout<<"\n\n \* \* \* \* PIZZA PARLOUR \* \* \* \* \n\n";

cout<<"\n 1. Add a Pizza \n 2. Display the Orders \n 3. Serve a pizza \n 4. Exit \n Enter your choice : ";

cin>>ch;

switch(ch)

{

case 1:

add();

break;

case 2:

display();

break;

case 3:

serve();

break;

case 4:

exit(0);

default:

cout<<"Invalid choice ";

check();

}

char ch1;

cout<<"\n Do you want to continue? ";

cin>>ch1;

if(ch1=='y'||ch1=='Y')

check();

}

};

int main()

{

pizza p1;

p1.check();

return 0;

}

OUTPUT

\* \* \* \* PIZZA PARLOUR \* \* \* \*

1. Add a Pizza

2. Display the Orders

3. Serve a pizza

4. Exit

Enter your choice : 1

Enter the Pizza ID: 21

Do you want to add another order ? Y

Enter the Pizza ID: 23

Do you want to add another order ? N

Do you want to continue? Y

\* \* \* \* PIZZA PARLOUR \* \* \* \*

1. Add a Pizza

2. Display the Orders

3. Serve a pizza

4. Exit

Enter your choice : 2

21<- 23

Do you want to continue? 3