Index

|  |  |  |
| --- | --- | --- |
| Pr. No. | Title | Page no. |
| 1 | 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 | 2 |
| 2 | Write a Python program to compute following operations on String:  a) To display word with the longest length  b) To determines the frequency of occurrence of particular character in the string  c) To check whether given string is palindrome or not  d) To display index of first appearance of the substring  e) To count the occurrences of each word in a given string | 3 |
| 3 | Write a python program to compute following computation on matrix:  a) Addition of two matrices  b) Subtraction of two matrices  c) Multiplication of two matrices  d) Transpose of a matrix | 5 |
| 4 | a) Write a pythonprogram 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.  b) 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 | 7 |
| 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 quick sort and display top five scores. | 9 |
| 6 | 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. | 10 |
| 7 | Write C++ program for storing appointment schedule for day. Appointments are booked randomly using linked list. Set start and end time and min and max duration for visit slot. Write functions for  a) Display free slots  b) Book appointment  c) Cancel appointment ( check validity, time bounds, availability)  d) Sort list based on time  e) Sort list based on time using pointer manipulation | 15 |
| 8 | 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. | 20 |
| 9 | 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. | 21 |
| 10 | 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 | 23 |
| 11 | 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 onedimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque | 27 |
| 12 | 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. | 31 |

PR 1

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

total\_students = int(input("Enter the number of students: "))  
score\_list = []  
absent\_list = []  
max\_marks = 0  
min\_marks = 0  
avg = 0  
  
for i in range(1, total\_students + 1):  
 marks = int(input(f"Enter the marks for student {i}: "))  
 if marks == -1:  
 name = input("Name of absent student: ")  
 absent\_list.append(name)  
 continue  
 else:  
 score\_list.append(marks)  
 avg = sum(score\_list) / total\_students  
 max\_marks = max(score\_list)  
 min\_marks = min(score\_list)  
 unique\_marks = set(score\_list)  
  
print("--------------------------------------------------------")  
print("The Avg Score of the Class:", avg)  
print("The Max Score of the Class:", max\_marks)  
print("The Min Score of the Class:", min\_marks)  
print("The Name of Absent Students are:", absent\_list)  
print("The Most of the students had got:", max(unique\_marks), "Marks")

Output:

Enter the number of students: 5

Enter the marks for student 1: 70

Enter the marks for student 2: 57

Enter the marks for student 3: 89

Enter the marks for student 4: 67

Enter the marks for student 5: 77

--------------------------------------------------------

The Avg Score of the Class: 72.0

The Max Score of the Class: 89

The Min Score of the Class: 57

The Name of Absent Students are: []

The Most of the students had got: 89 Marks

PR 2

Write a Python program to compute following operations on String:

a) To display word with the longest length

b) To determines the frequency of occurrence of particular character in the string

c) To check whether given string is palindrome or not

d) To display index of first appearance of the substring

e) To count the occurrences of each word in a given string

def longest\_word(sentence):  
 words = sentence.split()  
 longest = max(words, key=len)  
 return longest  
  
def char\_frequency(sentence, char):  
 return sentence.count(char)  
  
def is\_palindrome(word):  
 return word == word[::-1]  
  
def first\_appearance(substring, sentence):  
 return sentence.find(substring)  
  
def word\_occurrences(sentence):  
 words = sentence.split()  
 word\_count = {}  
 for word in words:  
 if word in word\_count:  
 word\_count[word] += 1  
 else:  
 word\_count[word] = 1  
 return word\_count  
  
*# Example string*input\_string = "Hello, how are you? Are you doing well? Let's check these operations."  
print("Input String : ",input\_string)  
  
*# Longest word*longest = longest\_word(input\_string)  
print("Longest word:", longest)  
  
*# Frequency of occurrence of a character*char\_to\_find = 'o'  
frequency = char\_frequency(input\_string, char\_to\_find)  
print(f"Frequency of '{char\_to\_find}':", frequency)  
  
*# Check palindrome*word\_to\_check = "madam"  
print(f"'{word\_to\_check}' is a palindrome:", is\_palindrome(word\_to\_check))  
  
*# First appearance of a substring*substring\_to\_find = "you"  
first\_index = first\_appearance(substring\_to\_find, input\_string)  
print(f"First appearance of '{substring\_to\_find}' at index:", first\_index)  
  
*# Word occurrences*word\_counts = word\_occurrences(input\_string)  
print("\nOccurrences of each word:")  
for word, count in word\_counts.items():  
 print(f"'{word}': {count}")

Output:

Input String : Hello, how are you? Are you doing well? Let's check these operations.

Longest word: operations.

Frequency of 'o': 7

'madam' is a palindrome: True

First appearance of 'you' at index: 15

Occurrences of each word:

'Hello,': 1

'how': 1

'are': 1

'you?': 1

'Are': 1

'you': 1

'doing': 1

'well?': 1

'Let's': 1

'check': 1

'these': 1

'operations.': 1

PR 3

Write a python program to compute following computation on matrix:

a) Addition of two matrices

b) Subtraction of two matrices

c) Multiplication of two matrices

d) Transpose of a matrix

def add\_matrices(matrix1, matrix2):  
 result = [[matrix1[i][j] + matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]  
 return result  
  
def subtract\_matrices(matrix1, matrix2):  
 result = [[matrix1[i][j] - matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]  
 return result  
  
def multiply\_matrices(matrix1, matrix2):  
 result = [[sum(matrix1[i][k] \* matrix2[k][j] for k in range(len(matrix2))) for j in range(len(matrix2[0]))] for i in range(len(matrix1))]  
 return result  
  
def transpose\_matrix(matrix):  
 result = [[matrix[j][i] for j in range(len(matrix))] for i in range(len(matrix[0]))]  
 return result  
  
*# Example matrices*matrix\_A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
matrix\_B = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]  
  
print("Matrix A:")  
for row in matrix\_A:  
 print(row)  
  
print("Matrix B:")  
for row in matrix\_B:  
 print(row)  
  
*# Addition*addition\_result = add\_matrices(matrix\_A, matrix\_B)  
print("\nAddition Result:")  
for row in addition\_result:  
 print(row)  
  
*# Subtraction*subtraction\_result = subtract\_matrices(matrix\_A, matrix\_B)  
print("\nSubtraction Result:")  
for row in subtraction\_result:  
 print(row)  
  
*# Multiplication*multiplication\_result = multiply\_matrices(matrix\_A, matrix\_B)  
print("\nMultiplication Result:")  
for row in multiplication\_result:  
 print(row)  
  
*# Transpose*transpose\_result\_A = transpose\_matrix(matrix\_A)  
print("\nTranspose of Matrix A:")  
for row in transpose\_result\_A:  
 print(row)

Output:

Matrix A:

[1, 2, 3]

[4, 5, 6]

[7, 8, 9]

Matrix B:

[9, 8, 7]

[6, 5, 4]

[3, 2, 1]

Addition Result:

[10, 10, 10]

[10, 10, 10]

[10, 10, 10]

Subtraction Result:

[-8, -6, -4]

[-2, 0, 2]

[4, 6, 8]

Multiplication Result:

[30, 24, 18]

[84, 69, 54]

[138, 114, 90]

Transpose of Matrix A:

[1, 4, 7]

[2, 5, 8]

[3, 6, 9]

PR 4

a) Write a pythonprogram 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.

b) 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

#a)

def linear\_search(arr, target):  
 for num in arr:  
 if num == target:  
 return True  
 return False  
  
def sentinel\_search(arr, target):  
 n = len(arr)  
 last = arr[n - 1]  
 arr[n - 1] = target *# Set sentinel value* i = 0  
 while arr[i] != target:  
 i += 1  
 arr[n - 1] = last *# Restore original array* if (i < n - 1) or (arr[n - 1] == target):  
 return True  
 return False  
  
*# Getting user-defined values*roll\_numbers = list(map(int, input("Enter roll numbers separated by space: ").split()))  
target\_roll = int(input("Enter the roll number to search: "))  
  
print("Linear Search:")  
if(linear\_search(roll\_numbers, target\_roll)==True):  
 print(f"Student with roll number {target\_roll} attended training program")  
else:  
 print(f"Student with roll number {target\_roll} not attended training program")  
  
*#print(f"Student with roll number {target\_roll} attended training program:", linear\_search(roll\_numbers, target\_roll))*print("\nSentinel Search:")  
if(sentinel\_search(roll\_numbers, target\_roll)==True):  
 print(f"Student with roll number {target\_roll} attended training program")  
else:  
 print(f"Student with roll number {target\_roll} not attended training program")  
*#print(f"Student with roll number {target\_roll} attended training program:", sentinel\_search(roll\_numbers, target\_roll))*

Output:

Enter roll numbers separated by space: 1 2 3 4 5 6 7 8 9 10

Enter the roll number to search: 5

Linear Search:

Student with roll number 5 attended training program

Sentinel Search:

Student with roll number 5 attended training program

b)

def binary\_search(arr, target):  
 left, right = 0, len(arr) - 1  
 while left <= right:  
 mid = left + (right - left) // 2  
 if arr[mid] == target:  
 return True  
 elif arr[mid] < target:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return False  
  
def fibonacci\_search(arr, target):  
 fib\_n\_2, fib\_n\_1 = 0, 1  
 fib\_sum = fib\_n\_1 + fib\_n\_2  
  
 while fib\_sum < len(arr):  
 fib\_n\_2 = fib\_n\_1  
 fib\_n\_1 = fib\_sum  
 fib\_sum = fib\_n\_1 + fib\_n\_2  
  
 offset = -1  
  
 while fib\_sum > 1:  
 i = min(offset + fib\_n\_2, len(arr) - 1)  
  
 if arr[i] < target:  
 fib\_sum = fib\_n\_1  
 fib\_n\_1 = fib\_n\_2  
 fib\_n\_2 = fib\_sum - fib\_n\_1  
 offset = i  
 elif arr[i] > target:  
 fib\_sum = fib\_n\_2  
 fib\_n\_1 = fib\_n\_1 - fib\_n\_2  
 fib\_n\_2 = fib\_sum - fib\_n\_1  
 else:  
 return True  
  
 if fib\_n\_1 and arr[offset + 1] == target:  
 return True  
  
 return False  
  
*# Getting user-defined values*sorted\_roll\_numbers = list(map(int, input("Enter sorted roll numbers separated by space: ").split()))  
target\_roll\_sorted = int(input("Enter the roll number to search in sorted list: "))  
  
print("Binary Search:")  
print("\nSentinel Search:")  
if(binary\_search(sorted\_roll\_numbers, target\_roll\_sorted)==True):  
 print(f"Student with roll number {target\_roll\_sorted} attended training program")  
else:  
 print(f"Student with roll number {target\_roll\_sorted} not attended training program")  
  
print("\nFibonacci Search:")  
if(fibonacci\_search(sorted\_roll\_numbers, target\_roll\_sorted)==True):  
 print(f"Student with roll number {target\_roll\_sorted} attended training program")  
else:  
 print(f"Student with roll number {target\_roll\_sorted} not attended training program")

Output

Enter sorted roll numbers separated by space: 1 2 3 4 5 6 7 8 9 10

Enter the roll number to search in sorted list: 1

Binary Search:

Sentinel Search:

Student with roll number 1 attended training program

Fibonacci Search:

Student with roll number 1 attended training program

PR 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 quick sort and display top five scores.

def quicksort(arr):  
 if len(arr) <= 1:  
 return arr  
 else:  
 pivot = arr[0]  
 less = [x for x in arr[1:] if x <= pivot]  
 greater = [x for x in arr[1:] if x > pivot]  
 return quicksort(less) + [pivot] + quicksort(greater)  
  
def display\_top\_scores(arr, n):  
 sorted\_arr = quicksort(arr)  
 top\_scores = sorted\_arr[-n:]  
 return top\_scores  
  
def get\_student\_percentages():  
 percentages = []  
 num\_students = int(input("Enter the number of students: "))  
 for i in range(num\_students):  
 percentage = float(input(f"Enter percentage for student {i + 1}: "))  
 percentages.append(percentage)  
 return percentages  
  
def main():  
 percentages = get\_student\_percentages()  
  
 *# Displaying the sorted array and top five scores* sorted\_percentages = quicksort(percentages)  
 print("\nSorted percentages in ascending order:", sorted\_percentages)  
  
 top\_five\_scores = display\_top\_scores(percentages, 5)  
 print("\nTop five scores:", top\_five\_scores)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Output:

Enter the number of students: 5

Enter percentage for student 1: 77.11

Enter percentage for student 2: 70.90

Enter percentage for student 3: 80.20

Enter percentage for student 4: 85.55

Enter percentage for student 5: 70.70

Sorted percentages in ascending order: [70.7, 70.9, 77.11, 80.2, 85.55]

Top five scores: [70.7, 70.9, 77.11, 80.2, 85.55]

PR 6

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.

#include <iostream>

using namespace std;

// Node structure to hold student information

struct Node {

int prn;

string name;

Node\* next;

};

class PinnacleClub {

private:

Node\* president; // First node for president

Node\* secretary; // Last node for secretary

int memberCount;

public:

PinnacleClub() {

president = NULL;

secretary = NULL;

memberCount = 0;

}

// Function to add a member at the end

void addMember(int prn, string name) {

Node\* newNode = new Node();

newNode->prn = prn;

newNode->name = name;

newNode->next = NULL;

if (president == NULL) {

president = newNode;

secretary = newNode;

} else {

secretary->next = newNode;

secretary = newNode;

}

memberCount++;

}

// Function to delete a member by PRN

void deleteMember(int prn) {

Node\* current = president;

Node\* prev = NULL;

// Search for the node with the given PRN

while (current != NULL && current->prn != prn) {

prev = current;

current = current->next;

}

// If the node is found, remove it

if (current != NULL) {

if (current == president) {

president = president->next;

} else if (current == secretary) {

secretary = prev;

secretary->next = NULL;

} else {

prev->next = current->next;

}

delete current;

memberCount--;

} else {

cout << "PRN not found. Could not delete member." << endl;

}

}

// Function to compute the total number of members

int getTotalMembers() {

return memberCount;

}

// Function to display all members

void displayMembers() {

Node\* temp = president;

while (temp != NULL) {

cout << "PRN: " << temp->prn << " Name: " << temp->name << endl;

temp = temp->next;

}

}

// Function to concatenate two lists

void concatenateLists(PinnacleClub& otherClub) {

if (otherClub.president == NULL) {

return; // Other list is empty, no need to concatenate

}

if (president == NULL) {

president = otherClub.president;

secretary = otherClub.secretary;

} else {

secretary->next = otherClub.president;

secretary = otherClub.secretary;

}

memberCount += otherClub.memberCount;

otherClub.president = NULL; // Clear the other list

otherClub.secretary = NULL;

otherClub.memberCount = 0;

}

};

int main() {

PinnacleClub club;

int prn;

string name;

int choice;

while (true) {

cout << "\n1. Add Member\n2. Delete Member\n3. Display Members\n4. Total Members\n5. Concatenate Lists\n6. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter PRN: ";

cin >> prn;

cout << "Enter Name: ";

cin >> name;

club.addMember(prn, name);

break;

case 2:

cout << "Enter PRN to delete: ";

cin >> prn;

club.deleteMember(prn);

break;

case 3:

cout << "\nClub Members:" << endl;

club.displayMembers();

break;

case 4:

cout << "Total Members: " << club.getTotalMembers() << endl;

break;

case 5:

// Functionality to concatenate lists not invoked from the menu for brevity

9 cout << "Concatenating lists..." << endl;

break;

case 6:

cout << "Exiting program. Goodbye!";

return 0;

default:

cout << "Invalid choice. Please try again." << endl;

}

}

return 0;

}

Output:

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 1

Enter PRN: 101

Enter Name: Satyam

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 1

Enter PRN: 102

Enter Name: Aditya

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 1

Enter PRN: 103

Enter Name: Vishal

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 3

Club Members:

PRN: 101 Name: Satyam

PRN: 102 Name: Aditya

PRN: 103 Name: Vishal

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 4

Total Members: 3

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 5

Concatenating lists...

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 2

Enter PRN to delete: 103

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 3

Club Members:

PRN: 101 Name: Satyam

PRN: 102 Name: Aditya

1. Add Member

2. Delete Member

3. Display Members

4. Total Members

5. Concatenate Lists

6. Exit

Enter your choice: 6

Exiting program. Goodbye!

--------------------------------

PR 7

Write C++ program for storing appointment schedule for day. Appointments are booked randomly using linked list. Set start and end time and min and max duration for visit slot. Write functions for

a) Display free slots

b) Book appointment

c) Cancel appointment ( check validity, time bounds, availability)

d) Sort list based on time

e) Sort list based on time using pointer manipulation

#include<iostream>

using namespace std;

int size; // No of Nodes or Appointments

struct SLL\_Node // Node Structure of each Appointment

{

int start;

int end;

int min;

int max;

int flag;

struct SLL\_Node \*next;

}\*head;

class App\_Shedule

{

public:

void create\_Shed();

void display\_Shed();

void book\_App();

void cancel\_App();

void sort\_App();

} A1;

int main()

{

int ch;

char ans;

do

{

cout<<"\n\n \*\*\* Menu \*\*\*";

cout<<"\n 1. Create Appointment Schedule";

cout<<"\n 2. Display Free Slots";

cout<<"\n 3. Book an Appointment";

cout<<"\n 4. Cancel an Appointment";

cout<<"\n 5. Sort slots based on Time";

cout<<"\n\n\t Enter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

A1.create\_Shed();

break;

case 2:

A1.display\_Shed();

break;

case 3:

A1.book\_App();

break;

case 4:

A1.cancel\_App();

break;

case 5:

A1.sort\_App();

break;

default:

cout<<"\n\t Wrong choice!!!";

}

cout<<"\n\n\t Do you wanna continue? (y/n) : ";

cin>>ans;

} while(ans == 'y');

}

void App\_Shedule::create\_Shed()

{

int i;

struct SLL\_Node \*temp, \*last;

head = NULL;

cout<<"\n\n\t How many Appointment Slots: ";

cin>>size;

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

{

temp = new(struct SLL\_Node); // Step 1: Dynamic Memory Allocation

cout<<"\n\n\t Enter Start Time: "; // Step 2: Assign Data & Address

cin>>temp->start;

cout<<"\n\t Enter End Time: ";

cin>>temp->end;

cout<<"\n\n\t Enter Minimum Duration: ";

cin>>temp->min;

cout<<"\n\t Enter Maximum Duration: ";

cin>>temp->max;

temp->flag = 0;

temp->next = NULL;

if(head == NULL)

{

head = temp;

last = head;

}

else

{

last->next = temp;

last = last->next;

}

}

}

void App\_Shedule::display\_Shed()

{

int cnt = 1;

struct SLL\_Node \*temp;

cout<<"\n\n\t \*\*\*\*Appointment Schdule\*\*\*\*";

cout<<"\n\n\t Srno.\tStart\tEnd\tMin\_Dur\tMax\_Dur\tStatus";

temp = head;

while(temp != NULL)

{

cout<<"\n\n\t "<<cnt;

cout<<"\t "<<temp->start;

cout<<"\t "<<temp->end;

cout<<"\t "<<temp->min;

cout<<"\t "<<temp->max;

if(temp->flag)

cout<<"\t-Booked-";

else

cout<<"\t--Free--";

temp = temp->next;

cnt++;

}

}

void App\_Shedule::book\_App()

{

int start;

struct SLL\_Node \*temp;

cout<<"\n\n\t Please enter Appointment time: ";

cin>>start;

temp = head;

while(temp != NULL)

{

if(start == temp->start)

{

if(temp->flag == 0)

{

cout<<"\n\n\t Appointment Slot is Booked!!!";

temp->flag = 1;

}

else

cout<<"\n\n\t Appointment Slot is not Available!!!";

}

temp = temp->next;

}

}

void App\_Shedule::cancel\_App()

{

int start;

struct SLL\_Node \*temp;

cout<<"\n\n\t Please enter Appointment time to Cancel: ";

cin>>start;

temp = head;

while(temp != NULL)

{

if(start == temp->start)

{

if(temp->flag == 1)

{

cout<<"\n\n\t Your Appointment Slot is Canceled!!!";

temp->flag = 0;

}

else

cout<<"\n\n\t Your Appointment was not Booked!!!";

}

temp = temp->next;

}

}

void App\_Shedule::sort\_App()

{

int i,j,val;

struct SLL\_Node \*temp;

for(i=0; i < size-1; i++)

{

temp = head;

while(temp->next != NULL)

{

if(temp->start > temp->next->start)

{

val = temp->start;

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

temp->next->start = val;

val = temp->end;

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

temp->next->end = val;

val = temp->min;

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

temp->next->min = val;

val = temp->max;

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

temp->next->max = val;

}

temp = temp->next;

}

}

cout<<"\n\n\t The Appointments got Sorted!!!";

}

Output:

\*\*\* Menu \*\*\*

1. Create Appointment Schedule

2. Display Free Slots

3. Book an Appointment

4. Cancel an Appointment

5. Sort slots based on Time

Enter your choice: 1

How many Appointment Slots: 3

Enter Start Time: 9

Enter End Time: 11

Enter Minimum Duration: 30

Enter Maximum Duration: 60

Enter Start Time: 12

Enter End Time: 14

Enter Minimum Duration: 40

Enter Maximum Duration: 70

Enter Start Time: 15

Enter End Time: 17

Enter Minimum Duration: 45

Enter Maximum Duration: 75

Do you wanna continue? (y/n) : y

\*\*\* Menu \*\*\*

1. Create Appointment Schedule

2. Display Free Slots

3. Book an Appointment

4. Cancel an Appointment

5. Sort slots based on Time

Enter your choice: 2

\*\*\*\*Appointment Schedule\*\*\*\*

Srno. Start End Min\_Dur Max\_Dur Status

1 9 11 30 60 --Free--

2 12 14 40 70 --Free--

3 15 17 45 75 --Free--

Do you wanna continue? (y/n) : y

\*\*\* Menu \*\*\*

1. Create Appointment Schedule

2. Display Free Slots

3. Book an Appointment

4. Cancel an Appointment

5. Sort slots based on Time

Enter your choice: 3

Please enter Appointment time: 12

Appointment Slot is Booked!!!

Do you wanna continue? (y/n) : y

\*\*\* Menu \*\*\*

1. Create Appointment Schedule

2. Display Free Slots

3. Book an Appointment

4. Cancel an Appointment

5. Sort slots based on Time

Enter your choice: 4

Please enter Appointment time to Cancel: 15

Your Appointment Slot is Canceled!!!

Do you wanna continue? (y/n) : y

\*\*\* Menu \*\*\*

1. Create Appointment Schedule

2. Display Free Slots

3. Book an Appointment

4. Cancel an Appointment

5. Sort slots based on Time

Enter your choice: 5

The Appointments got Sorted!!!

Do you wanna continue? (y/n) : n

PR 8

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.

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isBalanced(const string& expression) {

stack<char> s;

for (size\_t i = 0; i < expression.length(); ++i) {

char bracket = expression[i];

if (bracket == '(' || bracket == '[' || bracket == '{') {

s.push(bracket);

} else if (bracket == ')' || bracket == ']' || bracket == '}') {

if (s.empty()) {

return false;

}

char top = s.top();

s.pop();

if ((bracket == ')' && top != '(') ||

(bracket == ']' && top != '[') ||

(bracket == '}' && top != '{')) {

return false;

}

}

}

return s.empty();

}

int main() {

string expression;

cout << "Enter an expression with parentheses: ";

getline(cin, expression);

if (isBalanced(expression)) {

cout << "The expression has balanced parentheses." << endl;

} else {

cout << "The expression does not have balanced parentheses." << endl;

}

return 0;

}

Output:

Enter an expression with parentheses: (a \* (b + c) / d)

The expression has balanced parentheses.

PR 9

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.

#include <iostream>

#include <stack>

#include <string>

#include <cctype>

using namespace std;

int precedence(char op) {

if (op == '+' || op == '-') {

return 1;

} else if (op == '\*' || op == '/') {

return 2;

}

return -1;

}

string infixToPostfix(const string& infix) {

stack<char> s;

string postfix = "";

for (size\_t i = 0; i < infix.length(); ++i) {

if (isalnum(infix[i])) {

postfix += infix[i];

} else if (infix[i] == '(') {

s.push(infix[i]);

} else if (infix[i] == ')') {

while (!s.empty() && s.top() != '(') {

postfix += s.top();

s.pop();

}

if (!s.empty() && s.top() == '(') {

s.pop();

}

} else {

while (!s.empty() && precedence(infix[i]) <= precedence(s.top())) {

postfix += s.top();

s.pop();

}

s.push(infix[i]);

}

}

while (!s.empty()) {

postfix += s.top();

s.pop();

}

return postfix;

}

int evaluatePostfix(const string& postfix) {

stack<int> s;

for (size\_t i = 0; i < postfix.length(); ++i) {

if (isdigit(postfix[i])) {

s.push(postfix[i] - '0');

} else {

int operand2 = s.top();

s.pop();

int operand1 = s.top();

s.pop();

switch (postfix[i]) {

case '+': s.push(operand1 + operand2); break;

case '-': s.push(operand1 - operand2); break;

case '\*': s.push(operand1 \* operand2); break;

case '/': s.push(operand1 / operand2); break;

}

}

}

return s.top();

}

int main() {

string infix;

cout << "Enter infix expression: ";

cin >> infix;

string postfix = infixToPostfix(infix);

cout << "Postfix expression: " << postfix << endl;

int result = evaluatePostfix(postfix);

cout << "Evaluation result: " << result << endl;

return 0;

}

Output:

Enter infix expression: a+b\*c-(d/e+f)

Postfix expression: abc\*+de/f+-

PR 10

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

#include <iostream>

#include <queue>

using namespace std;

class JobQueue {

private:

queue<string> jobs;

public:

// Function to add a job to the queue

void addJob(const string& job) {

jobs.push(job);

cout << "Job '" << job << "' added to the queue." << endl;

}

// Function to delete a job from the queue

void deleteJob() {

if (jobs.empty()) {

cout << "Job queue is empty. No job to delete." << endl;

} else {

cout << "Job '" << jobs.front() << "' deleted from the queue." << endl;

jobs.pop();

}

}

void displayJobs() {

if (jobs.empty()) {

cout << "Job queue is empty." << endl;

} else {

cout << "Jobs in the queue:" << endl;

queue<string> temp = jobs;

while (!temp.empty()) {

cout << temp.front() << endl;

temp.pop();

}

}

}

};

int main() {

JobQueue jobQueue;

int choice;

string job;

do {

cout << "\n1. Add Job\n2. Delete Job\n3. Display Jobs\n4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter job to add: ";

cin.ignore(); // Clear buffer before getline

getline(cin, job);

jobQueue.addJob(job);

break;

case 2:

jobQueue.deleteJob();

break;

case 3:

jobQueue.displayJobs();

break;

case 4:

cout << "Exiting program. Goodbye!";

return 0;

default:

cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 4);

return 0;

}

Output:

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 1

Enter job to add: job1

Job 'job1' added to the queue.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 1

Enter job to add: job2

Job 'job2' added to the queue.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 1

Enter job to add: job3

Job 'job3' added to the queue.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 3

Jobs in the queue:

job1

job2

job3

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 2

Job 'job1' deleted from the queue.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 2

Job 'job2' deleted from the queue.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 2

Job 'job3' deleted from the queue.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 2

Job queue is empty. No job to delete.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 3

Job queue is empty.

1. Add Job

2. Delete Job

3. Display Jobs

4. Exit

Enter your choice: 4

Exiting program. Goodbye!

--------------------------------

PR 11

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 onedimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque

#include <iostream>

using namespace std;

class Deque {

private:

int\* arr;

int front;

int rear;

int capacity;

public:

Deque(int size) {

capacity = size + 1; // Extra space to distinguish between full and empty conditions

arr = new int[capacity];

front = 0;

rear = 0;

}

// Function to check if the deque is empty

bool isEmpty() {

return front == rear;

}

// Function to check if the deque is full

bool isFull() {

return (rear + 1) % capacity == front;

}

// Function to add an element at the front of the deque

void addFront(int element) {

if (isFull()) {

cout << "Deque is full. Cannot add element at the front." << endl;

return;

}

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

arr[front] = element;

cout << "Added element " << element << " at the front." << endl;

}

// Function to add an element at the rear of the deque

void addRear(int element) {

if (isFull()) {

cout << "Deque is full. Cannot add element at the rear." << endl;

return;

}

arr[rear] = element;

rear = (rear + 1) % capacity;

cout << "Added element " << element << " at the rear." << endl;

}

// Function to delete an element from the front of the deque

void deleteFront() {

if (isEmpty()) {

cout << "Deque is empty. Cannot delete element from the front." << endl;

return;

}

cout << "Deleted element " << arr[front] << " from the front." << endl;

front = (front + 1) % capacity;

}

// Function to delete an element from the rear of the deque

void deleteRear() {

if (isEmpty()) {

cout << "Deque is empty. Cannot delete element from the rear." << endl;

return;

}

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

cout << "Deleted element " << arr[rear] << " from the rear." << endl;

}

void displayElements() {

if (isEmpty()) {

cout << "Deque is empty. No elements to display." << endl;

return;

}

cout << "Elements in the Deque: ";

int index = front;

while (index != rear) {

cout << arr[index] << " ";

index = (index + 1) % capacity;

}

cout << endl;

}

};

int main() {

int size;

cout << "Enter the size of the Deque: ";

cin >> size;

Deque deque(size);

int choice, element;

do {

cout << "\n1. Add Front\n2. Add Rear\n3. Delete Front\n4. Delete Rear\n5. Display deque elements\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter element to add at the front: ";

cin >> element;

deque.addFront(element);

break;

case 2:

cout << "Enter element to add at the rear: ";

cin >> element;

deque.addRear(element);

break;

case 3:

deque.deleteFront();

break;

case 4:

deque.deleteRear();

break;

case 5:

cout << "Deque elements: ";

deque.displayElements();

break;

case 6:

cout << "Exiting program. Goodbye!";

return 0;

default:

cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 6);

return 0;

}

Output:

Enter the size of the Deque: 3

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 5

Deque elements: Deque is empty. No elements to display.

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 1

Enter element to add at the front: 3

Added element 3 at the front.

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 2

Enter element to add at the rear: 5

Added element 5 at the rear.

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 1

Enter element to add at the front: 7

Added element 7 at the front.

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 5

Deque elements: Elements in the Deque: 7 3 5

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 3

Deleted element 7 from the front.

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 4

Deleted element 5 from the rear.

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 5

Deque elements: Elements in the Deque: 3

1. Add Front

2. Add Rear

3. Delete Front

4. Delete Rear

5. Display deque elements

Enter your choice: 6

Exiting program. Goodbye!

PR 12

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.

#include <iostream>

using namespace std;

class PizzaParlor {

private:

int\* orders;

int front;

int rear;

int capacity;

int count;

public:

PizzaParlor(int size) {

capacity = size + 1; // Extra space to distinguish between full and empty conditions

orders = new int[capacity];

front = 0;

rear = 0;

count = 0;

}

bool isEmpty() {

return count == 0;

}

bool isFull() {

return count == capacity - 1;

}

void placeOrder(int orderNum) {

if (isFull()) {

cout << "Sorry, order queue is full. Cannot place order " << orderNum << "." << endl;

return;

}

orders[rear] = orderNum;

rear = (rear + 1) % capacity;

count++;

cout << "Order " << orderNum << " placed successfully." << endl;

}

void serveOrder() {

if (isEmpty()) {

cout << "No orders to serve." << endl;

return;

}

int servedOrder = orders[front];

front = (front + 1) % capacity;

count--;

cout << "Order " << servedOrder << " served." << endl;

}

void displayOrders() {

if (isEmpty()) {

cout << "No orders to display." << endl;

return;

}

cout << "Pending orders in the queue: ";

int idx = front;

while (idx != rear) {

cout << orders[idx] << " ";

idx = (idx + 1) % capacity;

}

cout << endl;

}

};

int main() {

int maxOrders;

cout << "Enter the maximum number of orders the pizza parlor can accept: ";

cin >> maxOrders;

PizzaParlor pizzaParlor(maxOrders);

int choice;

int orderNum;

do {

cout << "\n1. Place an Order\n2. Serve an Order\n3. Display Pending Orders\n4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

if (!pizzaParlor.isFull()) {

cout << "Enter the order number: ";

cin >> orderNum;

pizzaParlor.placeOrder(orderNum);

} else {

cout << "Sorry, order queue is full." << endl;

}

break;

case 2:

pizzaParlor.serveOrder();

break;

case 3:

pizzaParlor.displayOrders();

break;

case 4:

cout << "Exiting program. Goodbye!";

return 0;

default:

cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 4);

return 0;

}

Output:

Enter the maximum number of orders the pizza parlor can accept: 3

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 1

Enter the order number: 1

Order 1 placed successfully.

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 3

Pending orders in the queue: 1

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 1

Enter the order number: 3

Order 3 placed successfully.

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 1

Enter the order number: 4

Order 4 placed successfully.

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 1

Sorry, order queue is full.

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 2

Order 1 served.

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 3

Pending orders in the queue: 3 4

1. Place an Order

2. Serve an Order

3. Display Pending Orders

4. Exit

Enter your choice: 4

Exiting program. Goodbye!