DOMAIN WINTER CAMP WORKSHEET

DAY-3 (21/12/2024)

Student Name :- Pratham Kapoor **University ID :- 22BCS10732**

Branch :- B.E. (C.S.E.) **Section/Group :- FL_ 603-A**

Problem-1 :- Fibonacci numbers, commonly denoted by F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is, F(0) = 0, F(1) = 1, F(n) = F(n - 1) + F(n - 2), for n > 1. Given n, calculate F(n). **(Very Easy)**

Source Code

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

int main() {
   int Number;
   cout<<"Enter the Number :- ";
   cin>>Number;

   if(Number==0) {
      cout<<"\nFibonacci Number is 0";
   } else if(Number==1) {
      cout<<"\nFibonacci Number is 1";
   } else {
      int First=0,Second=1,Fibonacci;
      for(int Index=2;Index<=Number;Index++) {
         Fibonacci=First+Second;
         First=Second;
         Second=Fibonacci;
```

```
        }
        cout<<"\nThe Fibonacci Number is "<<Fibonacci;
    }

    return 0;
}
```

Output

```
Enter the Number :- 2

The Fibonacci Number is 1
```

Problem-2 :- Given the head of a singly linked list, reverse the list, and return the reversed list. **(Easy)**

Source Code

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

struct Node {
    int Value;
    Node* Next;
};

Node* ReverseList(Node* Head) {
    Node* Previous=NULL;
    Node* Current=Head;
    Node* Next=NULL;

    while(Current!=NULL) {
        Next=Current->Next;
```

```
        Current->Next=Previous;
        Previous=Current;
        Current=Next;
    }
    return Previous;
}

void PrintList(Node* Head) {
    cout<<"\nThe Reversed List is ";
    while(Head!=NULL) {
        cout<<Head->Value<<" ";
        Head=Head->Next;
    }
    cout<<endl;
}

void InsertAtEnd(Node*& Head, int Number) {
    Node* NewNode=new Node();
    NewNode->Value=Number;
    NewNode->Next=NULL;

    if(Head==NULL)
    {
        Head=NewNode;
    } else {
        Node* Temp=Head;
        while(Temp->Next!=NULL)
        {
            Temp=Temp->Next;
        }
        Temp->Next=NewNode;
    }
```

```
}

int main() {
   Node* Head=NULL;
   int N,Number;

   cout<<"Enter the Number of Elements in the List :- ";
   cin>>N;

   cout<<"Enter the Elements of the List :- ";
   for(int i=0;i<N;i++) {
      cin>>Number;
      InsertAtEnd(Head,Number);
   }

   Head=ReverseList(Head);
   PrintList(Head);

   return 0;
}
```

Output

```
Enter the Number of Elements in the List :- 5
Enter the Elements of the List :- 1 2 3 4 5

The Reversed List is 5 4 3 2 1
```

Problem-3:- You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list. You may assume the two numbers do not contain any leading zero, except the number 0 itself. **(Medium)**\

Source Code

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

struct Node {
   int Value;
   Node* Next;
};

Node* AddTwoNumbers(Node* L1, Node* L2) {
   Node* DummyNode=new Node();
   Node* Current=DummyNode;
   int Carry=0;

   while(L1!=NULL || L2!=NULL || Carry!=0) {
      int Sum=Carry;

      if(L1!=NULL) {
         Sum+=L1->Value;
         L1=L1->Next;
      }

      if(L2!=NULL) {
         Sum+=L2->Value;
         L2=L2->Next;
      }

      Carry=Sum/10;
      Node* NewNode=new Node();
      NewNode->Value=Sum%10;
      NewNode->Next=NULL;
```

```
        Current->Next=NewNode;
        Current=NewNode;
    }

    return DummyNode->Next;
}

void InsertAtEnd(Node*& Head, int Number) {
    Node* NewNode=new Node();
    NewNode->Value=Number;
    NewNode->Next=NULL;

    if(Head==NULL) {
        Head=NewNode;
    } else {
        Node* Temp=Head;
        while(Temp->Next!=NULL) {
            Temp=Temp->Next;
        }
        Temp->Next=NewNode;
    }
}

void PrintList(Node* Head) {
    while(Head!=NULL) {
        cout<<Head->Value<<" ";
        Head=Head->Next;
    }
    cout<<endl;
}

int main() {
```

```
    Node* L1=NULL;
    Node* L2=NULL;

    int N1,N2,Number;

    cout<<"Enter the Number of Elements in First List :- ";
    cin>>N1;

    cout<<"Enter the Elements of First List - ";
    for(int i=0;i<N1;i++)
   {
       cin>>Number;
       InsertAtEnd(L1,Number);
    }

    cout<<"\nEnter the Number of Elements in Second List :- ";
    cin>>N2;

    cout<<"Enter the Elements of Second List - ";
    for(int i=0;i<N2;i++)
   {
       cin>>Number;
       InsertAtEnd(L2,Number);
    }

    Node* Result=AddTwoNumbers(L1,L2);

    cout<<"\nThe Sum List is ";
    PrintList(Result);

    return 0;
}
```

Output

```
Enter the Number of Elements in First List :- 3
Enter the Elements of First List - 2 4 3

Enter the Number of Elements in Second List :- 3
Enter the Elements of Second List - 5 6 4

The Sum List is 7 0 8
```

Problem-4 :- Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where :- '?' Matches any single character. '*' Matches any sequence of characters (including empty sequence). The matching should cover the entire input string (not partial). **(Hard)**

Source Code

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

bool IsMatch(string S, string P) {
   int M=S.length();
   int N=P.length();
   vector<vector<bool>> DP(M+1,vector<bool>(N+1,false));

   DP[0][0]=true;

   for(int J=1;J<=N;J++) {
      if(P[J-1]=='*') {
         DP[0][J]=DP[0][J-1];
      }
   }
```

```
    for(int I=1;I<=M;I++)
   {
       for(int J=1;J<=N;J++) {
          if(P[J-1]=='?' || P[J-1]==S[I-1]) {
             DP[I][J]=DP[I-1][J-1];
          } else if(P[J-1]=='*') {
             DP[I][J]=DP[I-1][J] || DP[I][J-1];
          }
       }
    }

    return DP[M][N];
}

int main() {
    string S,P;

    cout<<"Enter the Input String :- ";
    cin>>S;

    cout<<"Enter the Pattern :- ";
    cin>>P;

    if(IsMatch(S,P)) {
       cout<<"\nTrue, the Pattern Matches the String";
    } else {
       cout<<"\nFalse, the Pattern Does Not Match the String";
    }

    return 0;
}
```

Output

```
Enter the Input String :- aa
Enter the Pattern :- a

False, the Pattern Does Not Match the String
```

Problem-5 :- Special binary strings are binary strings with the following two properties :- The number of 0's is equal to the number of 1's. Every prefix of the binary string has at least as many 1's as 0's. You are given a special binary string s. A move consists of choosing two consecutive, non-empty, special substrings of s, and swapping them. Two strings are consecutive if the last character of the first string is exactly one index before the first character of the second string. Return the lexicographically largest resulting string possible after applying the mentioned operations on the string. **(Very Hard)**

Source Code

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

string MakeLargestSpecial(string S)
{
   vector<string> Substrings;
   int Count=0,Start=0;

   for(int I=0;I<S.size();I++)
   {
      if(S[I]=='1') Count++;
```

```
        else Count--;

        if(Count==0) {
            string      Substring="1"+MakeLargestSpecial(S.substr(Start+1,I-Start-
1))+"0";
            Substrings.push_back(Substring);
            Start=I+1;
        }
    }

    sort(Substrings.rbegin(),Substrings.rend());
    string Result="";
    for(string &Sub:Substrings)
    {
        Result+=Sub;
    }
    return Result;
}

int main()
{
    string S;

    cout<<"Enter the Special Binary String :- ";
    cin>>S;

    string Result=MakeLargestSpecial(S);

    cout<<"\nThe Lexicographically Largest String is "<<Result;

    return 0;
}
```

Output

```
Enter the Special Binary String :- 11011000

The Lexicographically Largest String is 11100100
```

Problem-6 :- You are given the firstname and the lastname of a person on two different lines. Your task is to read them and print the following using function :- Hello firstname lastname! You just delved into function. **(Very Easy)**

Source Code

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

void PrintFullName(string FirstName, string LastName) {
   cout<<"\nHello "<<FirstName<<" "<<LastName<<"! You just Delved into Function";
}

int main() {
   string FirstName, LastName;

   cout<<"Enter the First Name :- ";
   cin>>FirstName;

   cout<<"Enter the Last Name :- ";
   cin>>LastName;

   PrintFullName(FirstName, LastName);

   return 0; }
```

Output

```
Enter the First Name :- Ross
Enter the Last Name :- Taylor

Hello Ross Taylor! You just Delved into Function
```

Problem-7 :- Given an integer array nums, return the greatest common divisor of the smallest number and largest number in nums. The greatest common divisor of two numbers is the largest positive integer that evenly divides both numbers. **(Easy)**

Source Code

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

int GCD(int a, int b)
{
   while(b!=0) {
      int temp=b;
      b=a%b;
      a=temp;
   }
   return a;
}

int main()
{
   vector<int> nums;
   int N, Number;
```

```
    cout<<"Enter the Number of Elements in the Array :- ";
    cin>>N;

    cout<<"Enter the Elements of the Array :- ";
    for(int i=0; i<N; i++) {
        cin>>Number;
        nums.push_back(Number);
    }

    int Smallest=*min_element(nums.begin(), nums.end());
    int Largest=*max_element(nums.begin(), nums.end());

    int Result=GCD(Smallest, Largest);
    cout<<"\nThe Greatest Common Divisor of "<<Smallest<<" and "<<Largest<<" is "<<Result;

    return 0;
}
```

Output

```
Enter the Number of Elements in the Array :- 5
Enter the Elements of the Array :- 2 5 6 9 10

The Greatest Common Divisor of 2 and 10 is 2
```

Problem-8 :- You are working on building a text editor application. One of the features you're focusing on is a "word suggestion" system. The editor needs to identify the longest sequence of characters typed without repeating any letters to suggest potential words or phrases. To accomplish this, you must efficiently find the length of the longest substring of unique characters as the user types. **(Medium)**

Source Code

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

int LengthOfLongestSubstring(string s) {
    unordered_map<char, int> charIndex;
    int maxLength = 0;
    int start = 0;

    for(int end = 0; end < s.size(); end++) {
        if(charIndex.find(s[end]) != charIndex.end()) {
            start = max(start, charIndex[s[end]] + 1);
        }
        charIndex[s[end]] = end;
        maxLength = max(maxLength, end - start + 1);
    }

    return maxLength;
}

int main() {
    string s;

    cout<<"Enter the String :- ";
    cin>>s;

    int result = LengthOfLongestSubstring(s);
    cout<<"\nThe Length of the Longest Substring Without Repeating Characters is "<<result;
```

```
    return 0;
}
```

Output

```
Enter the String :- abcabcabc

The Length of the Longest Substring Without Repeating Characters is 3
```

Problem-9 :- You are developing a financial analysis application where users input daily stock price changes in an array. Your task is to determine the maximum profit or loss achievable over a contiguous period, considering that the profit/loss for a period is calculated by multiplying the daily percentage changes. This problem is critical for finding optimal periods to make decisions for investments or short-term trading. **(Hard)**

Source Code

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

int MaxProduct(vector<int>& nums)
{
    int N = nums.size();
    int MaxProduct = nums[0];
    int MinProduct = nums[0];
    int Result = nums[0];

    for(int i = 1; i < N; i++)
   {
        if(nums[i] < 0) {
```

```
            swap(MaxProduct, MinProduct);
        }

        MaxProduct = max(nums[i], MaxProduct * nums[i]);
        MinProduct = min(nums[i], MinProduct * nums[i]);

        Result = max(Result, MaxProduct);
    }

    return Result;
}

int main()
{
    vector<int> nums;
    int N, Number;

    cout<<"Enter the Number of Elements in the Array :- ";
    cin>>N;

    cout<<"Enter the Elements of the Array - ";
    for(int i = 0; i < N; i++)
    {
        cin>>Number;
        nums.push_back(Number);
    }

    int Result = MaxProduct(nums);
    cout<<"\nThe Maximum Product of Contiguous Subarray is "<<Result;

    return 0;
}
```

Output

```
Enter the Number of Elements in the Array :- 4
Enter the Elements of the Array - 2 3 -2 4

The Maximum Product of Contiguous Subarray is 6
```

Problem-10 :- There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings. You are giving candies to these children subjected to the following requirements :- Each child must have at least one candy. Children with a higher rating get more candies than their neighbours. Return the minimum number of candies you need to have to distribute the candies to the children. **(Very Hard)**

Source Code

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

int Candy(vector<int>& ratings)
{
   int N = ratings.size();
   vector<int> candies(N, 1);

   for(int i = 1; i < N; i++)
  {
      if(ratings[i] > ratings[i-1])
     {
         candies[i] = candies[i-1] + 1;
      }
   }
```

```
    for(int i = N - 2; i >= 0; i--) {
        if(ratings[i] > ratings[i+1]) {
            candies[i] = max(candies[i], candies[i+1] + 1);
        }
    }

    int Result = 0;
    for(int i = 0; i < N; i++) {
        Result += candies[i];
    }

    return Result;
}

int main() {
    vector<int> ratings;
    int N, Rating;

    cout<<"Enter the Number of Children :- ";
    cin>>N;

    cout<<"Enter the Ratings of the Children :- ";
    for(int i = 0; i < N; i++) {
        cin>>Rating;
        ratings.push_back(Rating);
    }

    int Result = Candy(ratings);
    cout<<"\nThe Minimum Number of Candies Required are "<<Result;

    return 0; }
```

Output

```
Enter the Number of Children :- 3
Enter the Ratings of the Children :- 1 0 2

The Minimum Number of Candies Required are 5
```