DOMAIN WINTER CAMP

Department of Computer Science and Engineering

Name:- Aman Pal UID:- 22BCS10188 Section:- 22KPIT-901-A

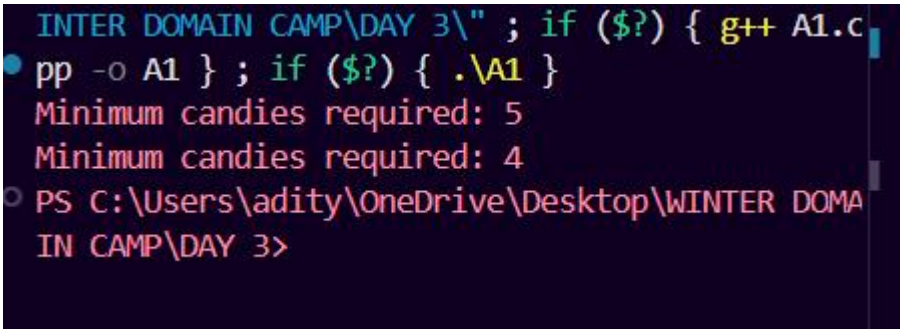
DAY-3

Q.1. 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 neighbors. Return the minimum number of candies you need to have to distribute the candies to the children. in cpp

Program Code:-

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

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

    // Left to right pass
    for (int i = 1; i < n; ++i)
    {
        if (ratings[i] > ratings[i - 1])
        {
            candies[i] = candies[i - 1] + 1;
        }
    }

    // Right to left pass
    for (int i = n - 2; i >= 0; --i)
    {
        if (ratings[i] > ratings[i + 1])
        {
            candies[i] = max(candies[i], candies[i + 1] + 1);
        }
    }

    // Calculate the total number of candies
    int totalCandies = 0;
    for (int candy : candies)
    {
        totalCandies += candy;
    }

    return totalCandies;
}

int main()
{
    vector<int> ratings = {1, 0, 2};
    cout << "Minimum candies required: " << candy(ratings) << endl;

    ratings = {1, 2, 2};
    cout << "Minimum candies required: " << candy(ratings) << endl;

    return 0;
```

Output:-


INTER DOMAIN CAMP\DAY 3\" ; if ($?) { g++ A1.c
pp -o A1 } ; if ($?) { .\A1 }
Minimum candies required: 5
Minimum candies required: 4
PS C:\Users\adity\OneDrive\Desktop\WINTER DOMA
IN CAMP\DAY 3>

Q.2.

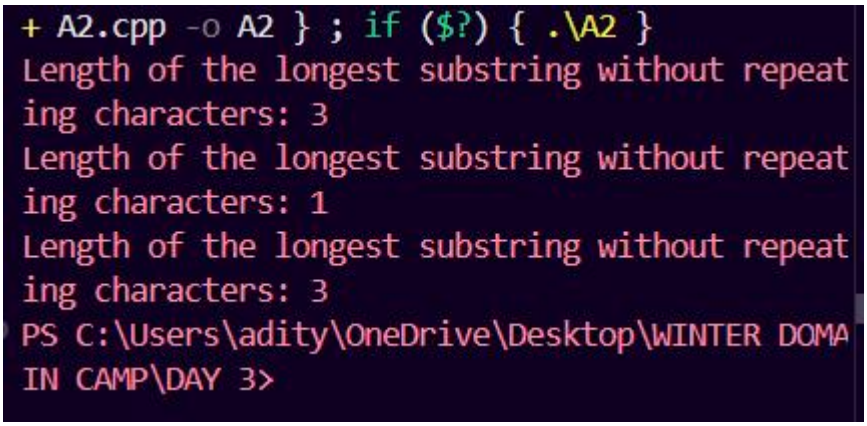
Longest Substring Without Repeating Characters

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.

Program Code:-

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

int lengthOfLongestSubstring(string s)
{
    unordered_set<char> charSet;
    int left = 0, maxLength = 0;

    for (int right = 0; right < s.length(); ++right)
    {
        // If the character is already in the set, remove characters from the left
        while (charSet.find(s[right]) != charSet.end())
        {
            charSet.erase(s[left]);
            ++left;
        }

        // Add the current character to the set
        charSet.insert(s[right]);

        // Update the maximum length
        maxLength = max(maxLength, right - left + 1);
    }

    return maxLength;
}

int main()
{
    string input = "abcabcbb";
    cout << "Length of the longest substring without repeating characters: " << lengthOfLongestSubstring(input) << endl;

    input = "bbbbb";
    cout << "Length of the longest substring without repeating characters: " << lengthOfLongestSubstring(input) << endl;

    input = "pwwkew";
    cout << "Length of the longest substring without repeating characters: " << lengthOfLongestSubstring(input) << endl;

    return 0;
}
```

Output:-



Q.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.

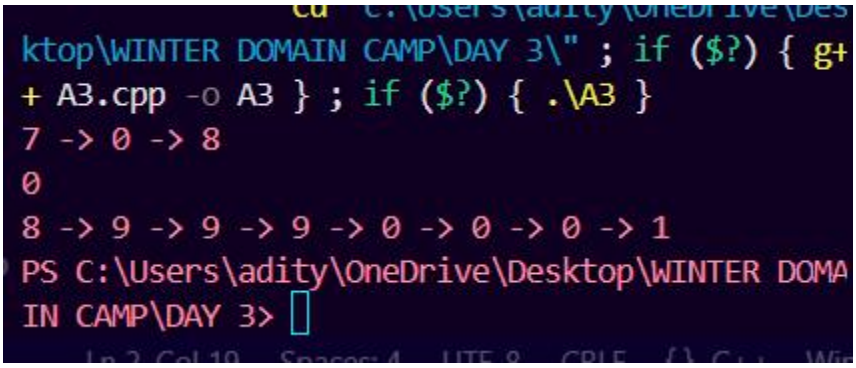
Program code:-

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

struct ListNode
{
    int val;
    ListNode *next;
    ListNode() : val(0), next(nullptr) {}
    ListNode(int x) : val(x), next(nullptr) {}
    ListNode(int x, ListNode *next) : val(x), next(next) {}
};

class Solution
{
public:
    ListNode *addTwoNumbers(ListNode *l1, ListNode *l2)
    {
        ListNode *dummyHead = new ListNode(0);
        ListNode *current = dummyHead;
        int carry = 0;

        while (l1 != nullptr || l2 != nullptr || carry != 0)
        {
            int sum = carry;

            if (l1 != nullptr)
            {
                sum += l1->val;
                l1 = l1->next;
            }

            if (l2 != nullptr)
            {
                sum += l2->val;
                l2 = l2->next;
            }

            carry = sum / 10;
            current->next = new ListNode(sum % 10);
            current = current->next;
        }

        return dummyHead->next;
    }
};
```

```cpp
ListNode *createLinkedList(const vector<int> &nums)
{
    ListNode *head = nullptr;
    ListNode *tail = nullptr;

    for (int num : nums)
    {
        ListNode *newNode = new ListNode(num);
        if (!head)
        {
            head = newNode;
            tail = head;
        }
        else
        {
            tail->next = newNode;
            tail = tail->next;
        }
    }

    return head;
}

void printLinkedList(ListNode *head)
{
    while (head)
    {
        cout << head->val;
        if (head->next)
            cout << " -> ";
        head = head->next;
    }
    cout << endl;
}

int main()
{
    Solution solution;

    ListNode *l1 = createLinkedList({2, 4, 3});
    ListNode *l2 = createLinkedList({5, 6, 4});
    ListNode *result = solution.addTwoNumbers(l1, l2);

    printLinkedList(result);

    l1 = createLinkedList({0});
    l2 = createLinkedList({0});
```

```cpp
int main()
{
    Solution solution;

    ListNode *l1 = createLinkedList({2, 4, 3});
    ListNode *l2 = createLinkedList({5, 6, 4});
    ListNode *result = solution.addTwoNumbers(l1, l2);

    printLinkedList(result);

    l1 = createLinkedList({0});
    l2 = createLinkedList({0});
    result = solution.addTwoNumbers(l1, l2);

    printLinkedList(result);

    l1 = createLinkedList({9, 9, 9, 9, 9, 9, 9});
    l2 = createLinkedList({9, 9, 9, 9});
    result = solution.addTwoNumbers(l1, l2);

    printLinkedList(result);

    return 0;
}
```

Output:-

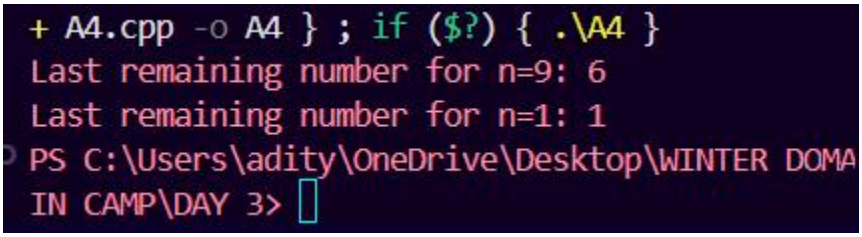


Q.4. Elimination game using cpp.

Program Code:-

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

class Solution
{
public:
    int lastRemaining(int n)
    {
        bool leftToRight = true;
        int remaining = n;
        int step = 1;
        int head = 1;

        while (remaining > 1)
        {
            if (leftToRight || remaining % 2 == 1)
            {
                head += step;
            }
            remaining /= 2;
            step *= 2;
            leftToRight = !leftToRight;
        }

        return head;
    }
};

int main()
{
    Solution solution;

    // Test cases for Elimination Game
    cout << "Last remaining number for n=9: " << solution.lastRemaining(9) << endl; // Output: 6
    cout << "Last remaining number for n=1: " << solution.lastRemaining(1) << endl; // Output: 1

    return 0;
}
```

Output:-


+ A4.cpp -o A4 } ; if ($?) { .\A4 }
Last remaining number for n=9: 6
Last remaining number for n=1: 1
PS C:\Users\adity\OneDrive\Desktop\WINTER DOMA
IN CAMP\DAY 3>

Q.5. Design a c++ program on

Regular Expression Matching

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '*' where:

'.' Matches any single character.

'*' Matches zero or more of the preceding element.

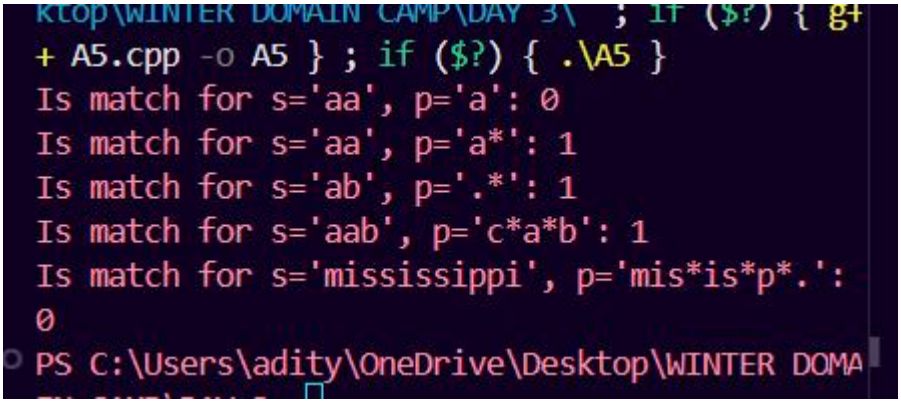
The matching should cover the entire input string (not partial).

Program Code:-

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

class Solution
{
public:
    bool isMatch(string s, string p)
    {
        int m = s.length(), n = p.length();
        bool dp[m + 1][n + 1];

        // Initialize DP table
        for (int i = 0; i <= m; ++i)
        {
            for (int j = 0; j <= n; ++j)
            {
                dp[i][j] = false;
            }
        }
        dp[0][0] = true;

        // Handle patterns with *
        for (int j = 1; j <= n; ++j)
        {
            if (p[j - 1] == '*')
            {
                dp[0][j] = dp[0][j - 2];
            }
        }

        // Fill the DP table
        for (int i = 1; i <= m; ++i)
        {
            for (int j = 1; j <= n; ++j)
            {
                if (p[j - 1] == s[i - 1] || p[j - 1] == '.')
                {
                    dp[i][j] = dp[i - 1][j - 1];
                }
                else if (p[j - 1] == '*')
```

```
                {
                    dp[i][j] = dp[i][j - 2] || (dp[i - 1][j] && (s[i - 1] == p[j - 2] || p[j - 2] == '.'));
                }
            }
        }

        return dp[m][n];
    }
};

int main()
{
    Solution solution;

    // Test cases for Regular Expression Matching
    cout << "Is match for s='aa', p='a': " << solution.isMatch("aa", "a") << endl;
    cout << "Is match for s='aa', p='a*': " << solution.isMatch("aa", "a*") << endl;
    cout << "Is match for s='ab', p='.*': " << solution.isMatch("ab", ".*") << endl;
    cout << "Is match for s='aab', p='c*a*b': " << solution.isMatch("aab", "c*a*b") << endl;
    cout << "Is match for s='mississippi', p='mis*is*p*.': " << solution.isMatch("mississippi", "mis*is*p*.")

    return 0;
}
```

Output:-


ktop\WINTER DOMAIN CAMP\DAY 3\ ; if ($?) { g+
+ A5.cpp -o A5 } ; if ($?) { .\A5 }
Is match for s='aa', p='a': 0
Is match for s='aa', p='a*': 1
Is match for s='ab', p='.*': 1
Is match for s='aab', p='c*a*b': 1
Is match for s='mississippi', p='mis*is*p*.':
0
PS C:\Users\adity\OneDrive\Desktop\WINTER DOMA