

WINTER DOMAIN CAMP

NAME: Vikram Kumar

SECTION: KPIT-901-B

UID:22BCS12220

DATE: 23/12/2024

Q1.Fibonacci series using recursion(Easy)

Sol.

```
1 //fibonacci series using recursion
2 #include <iostream>
3 using namespace std;
4 int fibonacci(int n) {
5     if (n == 0) return 0;
6     if (n == 1) return 1;
7     return fibonacci(n - 1) + fibonacci(n - 2);
8 }
9
10 int main() {
11     int n;
12     cout << "Enter the value of n: ";
13     cin >> n;
14
15     cout << "Fibonacci number F(" << n << ") is: " << fibonacci(n) << endl;
16     return 0;
17 }
```

OUTPUT:

```
Enter the value of n: 7
Fibonacci number F(7) is: 13
...Program finished with exit
```

## Q2. Merge two sorted links(medium).

```
1 //merge two sorted links
2 #include <iostream>
3 #include<vector>
4 using namespace std;
5
6 struct ListNode {
7     int val;
8     ListNode* next;
9     ListNode() : val(0), next(nullptr) {}
10    ListNode(int x) : val(x), next(nullptr) {}
11    ListNode(int x, ListNode* next) : val(x), next(next) {}
12 };
13
14 ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
15     if (!list1) return list2;
16     if (!list2) return list1;
17     if (list1->val < list2->val) {
18         list1->next = mergeTwoLists(list1->next, list2);
19         return list1;
20     } else {
21         list2->next = mergeTwoLists(list1, list2->next);
22         return list2;
23     }
24 }
25
26 ListNode* createLinkedList(const vector<int>& values) {
27     if (values.empty()) return nullptr;
28     ListNode* head = new ListNode(values[0]);
29     ListNode* current = head;
30     for (size_t i = 1; i < values.size(); ++i) {
31         current->next = new ListNode(values[i]);
32         current = current->next;
33     }
34     return head;
35 }
36
37 void printLinkedList(ListNode* head) {
38     while (head) {
39         cout << head->val;
40         if (head->next) cout << " -> ";
41         head = head->next;
42     }
```

```
43     cout << endl;
44 }
45
46 int main() {
47     vector<int> list1Values = {1, 2, 4};
48     vector<int> list2Values = {1, 3, 4};
49     ListNode* list1 = createLinkedList(list1Values);
50     ListNode* list2 = createLinkedList(list2Values);
51     ListNode* mergedList = mergeTwoLists(list1, list2);
52     cout << "Merged List: ";
53     printLinkedList(mergedList);
54
55     return 0;
56 }
```

OUTPUT:

```
Merged List: 1 -> 1 -> 2 -> 3 -> 4 -> 4  
  
...Program finished with exit code 0
```

Q3. Add two numbers(medium).

Sol.

```
1 //Add two numbers  
2 #include <iostream>  
3 #include<vector>  
4 using namespace std;  
5  
6 struct ListNode {  
7     int val;  
8     ListNode* next;  
9     ListNode() : val(0), next(nullptr) {}  
10    ListNode(int x) : val(x), next(nullptr) {}  
11    ListNode(int x, ListNode* next) : val(x), next(next) {}  
12 };  
13 ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {  
14     ListNode* dummyHead = new ListNode();  
15     ListNode* current = dummyHead;  
16     int carry = 0;  
17  
18     while (l1 || l2 || carry) {  
19         int sum = carry;  
20         if (l1) {  
21             sum += l1->val;  
22             l1 = l1->next;  
23         }  
24  
25         if (l2) {  
26             sum += l2->val;  
27             l2 = l2->next;  
28         }  
29  
30         carry = sum / 10;  
31         current->next = new ListNode(sum % 10);  
32         current = current->next;  
33     }  
34  
35     return dummyHead->next;  
36 }  
37 ListNode* createLinkedList(const vector<int>& values) {  
38     if (values.empty()) return nullptr;
```

```

39     ListNode* head = new ListNode(values[0]);
40     ListNode* current = head;
41     for (size_t i = 1; i < values.size(); ++i) {
42         current->next = new ListNode(values[i]);
43         current = current->next;
44     }
45     return head;
46 }
47 void printLinkedList(ListNode* head) {
48     while (head) {
49         cout << head->val;
50         if (head->next) cout << " , ";
51         head = head->next;
52     }
53     cout << endl;
54 }
55
56 int main() {
57     vector<int> l1Values = {2, 4, 3};
58     vector<int> l2Values = {5, 6, 4};
59     ListNode* l1 = createLinkedList(l1Values);
60     ListNode* l2 = createLinkedList(l2Values);
61     ListNode* result = addTwoNumbers(l1, l2);
62     cout << "Result: ";
63     printLinkedList(result);
64
65     return 0;
66 }

```

OUTPUT:

```

Result: 7 , 0 , 8
Program finish

```

## Q4.Elimination game hard)

Sol.

```

1 //Elimination game
2 #include <iostream>
3 using namespace std;
4
5 int lastRemaining(int n) {
6     int head = 1;
7     int step = 1;
8     bool left = true;
9     int remaining = n;
10
11     while (remaining > 1) {
12         if (left || remaining % 2 == 1) {
13             head += step;
14         }
15         step *= 2;
16         remaining /= 2;
17         left = !left;
18     }
19
20     return head;
21 }
22
23 int main() {
24     int n;
25
26     cout << "Enter n: ";
27     cin >> n;
28
29     cout << "Last remaining number is: " << lastRemaining(n) << endl;
30
31     return 0;
32 }

```

OUTPUT:

```

Enter n: 5
Last remaining number is: 2

```

## Q5. Regular expression matching.(hard)

Sol.

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  using namespace std;
5
6  bool isMatch(string s, string p) {
7      int m = s.size(), n = p.size();
8      vector<vector<bool>> dp(m + 1, vector<bool>(n + 1, false));
9      dp[0][0] = true;
10     for (int j = 2; j <= n; ++j) {
11         if (p[j - 1] == '*') {
12             dp[0][j] = dp[0][j - 2];
13         }
14     }
15     for (int i = 1; i <= m; ++i) {
16         for (int j = 1; j <= n; ++j) {
17             if (p[j - 1] == s[i - 1] || p[j - 1] == '.') {
18                 dp[i][j] = dp[i - 1][j - 1];
19             } else if (p[j - 1] == '*') {
20                 dp[i][j] = dp[i][j - 2];
21                 if (p[j - 2] == s[i - 1] || p[j - 2] == '.') {
22                     dp[i][j] = dp[i][j] || dp[i - 1][j];
23                 }
24             }
25         }
26     }
27     return dp[m][n];
28 }
29
30 int main() {
31     // Test cases
32     string s, p;
33     cout << "Enter string s: ";
34     cin >> s;
35     cout << "Enter pattern p: ";
36     cin >> p;
```

```
39     if (isMatch(s, p)) {
40         cout << "The string \"" << s << "\" matches the pattern \"" << p << "\"." << endl;
41     } else {
42         cout << "The string \"" << s << "\" does not match the pattern \"" << p << "\"." << endl;
43     }
44     return 0;
45 }
46 }
```

OUTPUT:

```
Enter string s: aa
Enter pattern p: a*
The string "aa" matches the pattern "a*".

...Program finished with exit code 0
```