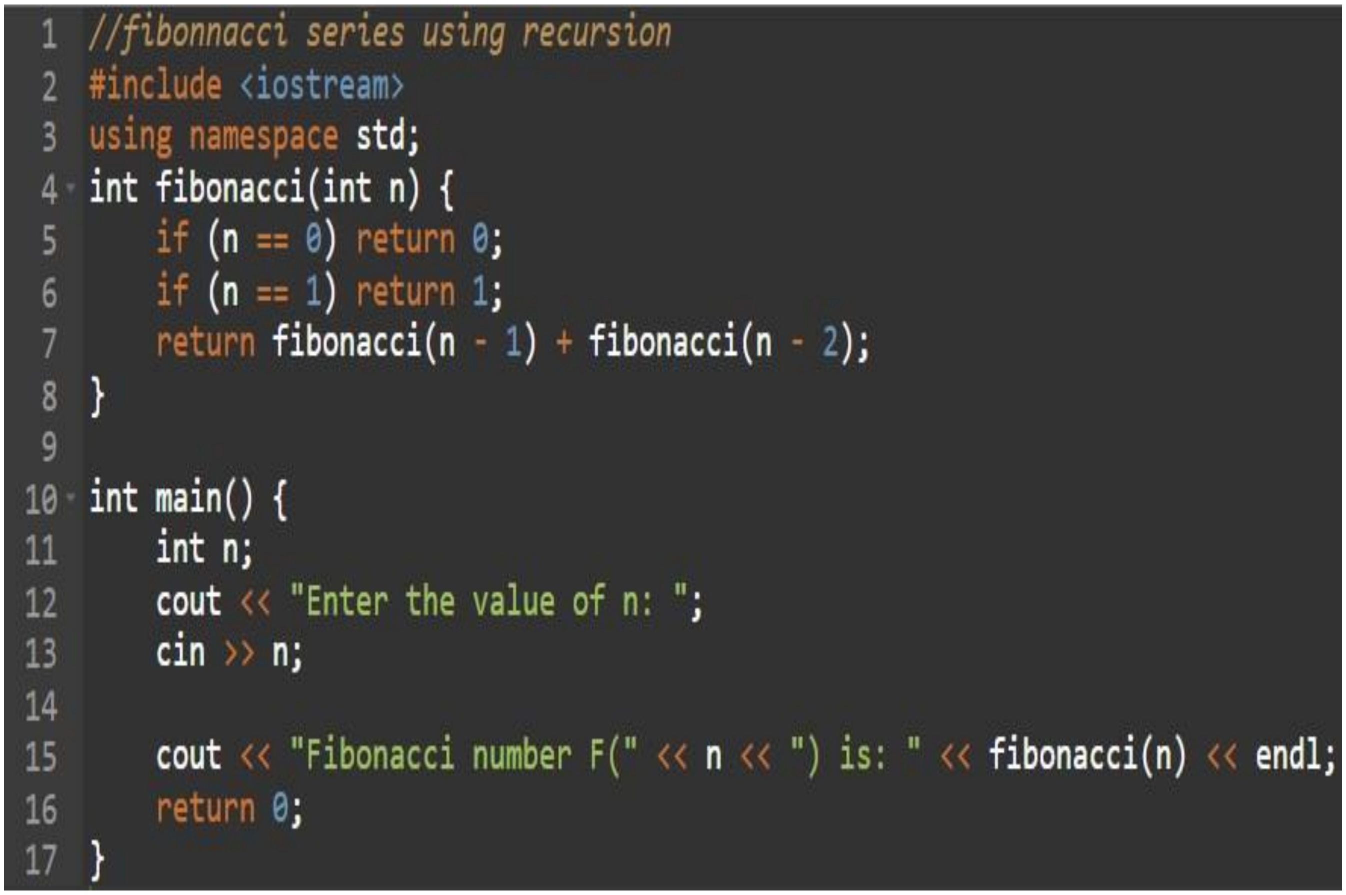
WINTER DOMAIN CAMP

NAME:Aparajita Garbyal **UID:22BCS11904**

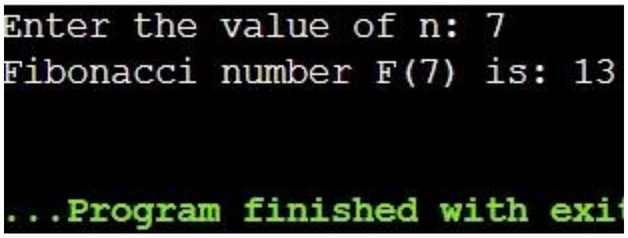
SECTION: KPIT-901-B **DATE: 23/12/2024**

Q1.Fibonnacci series using recursion(Easy) Sol.



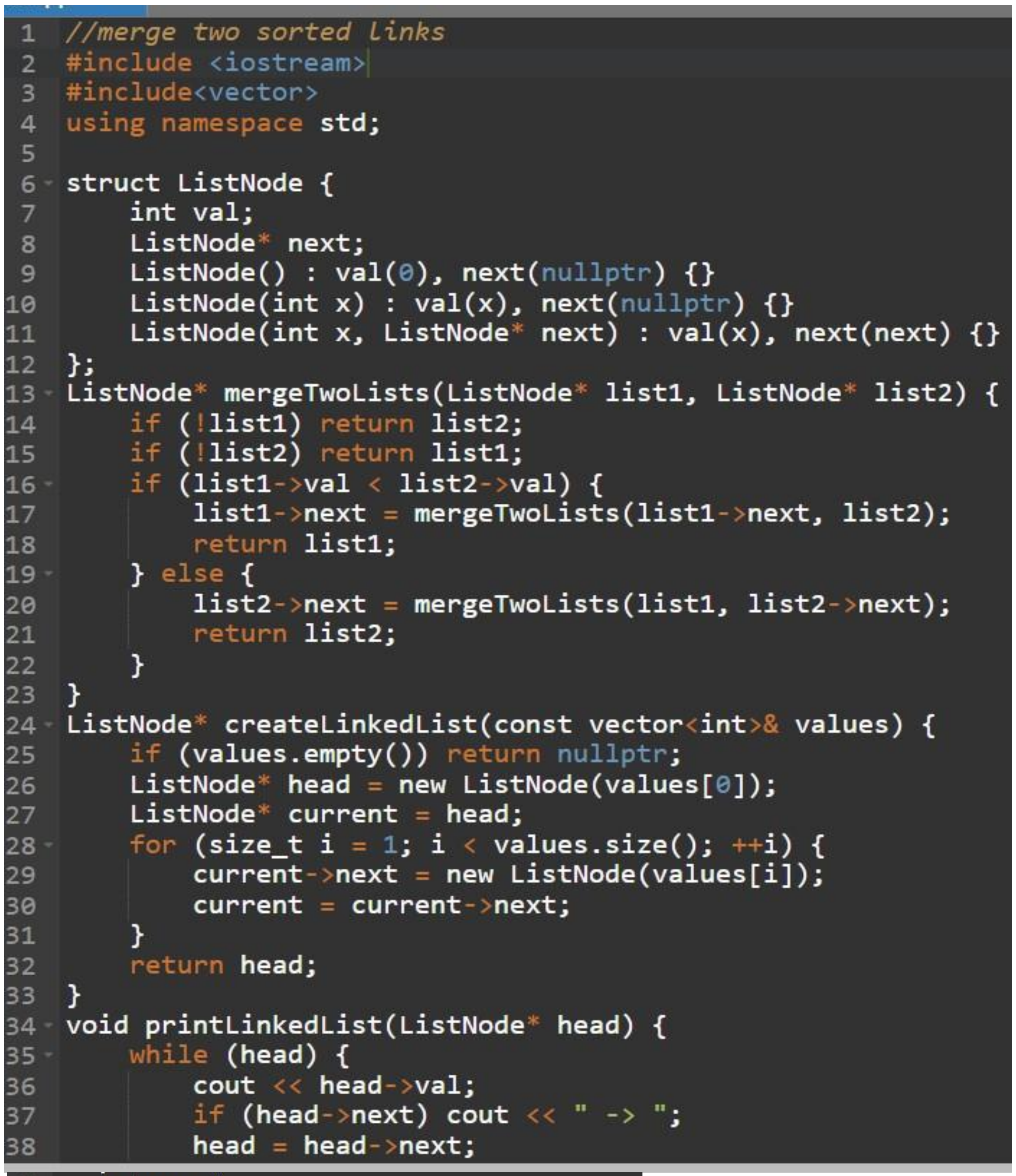
```cpp
//fibonnacci series using recursion
#include <iostream>
using namespace std;
int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n;
    cout << "Enter the value of n: ";
    cin >> n;

    cout << "Fibonacci number F(" << n << ") is: " << fibonacci(n) << endl;
    return 0;
}
```

OUTPUT:

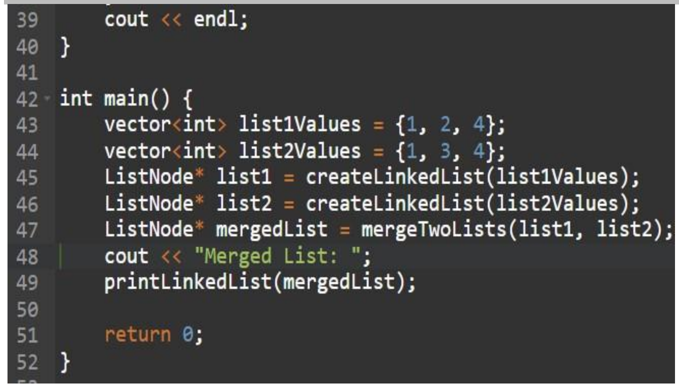


```
Enter the value of n: 7
Fibonacci number F(7) is: 13


...Program finished with exi
```

Q2. Merge two sorted links(medium).

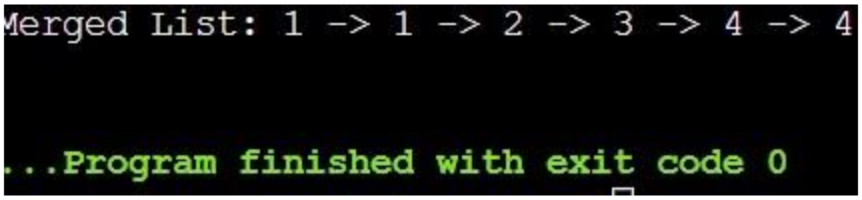


```cpp
//merge two sorted links
#include <iostream>
#include<vector>
using namespace std;

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) {}
};
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
    if (!list1) return list2;
    if (!list2) return list1;
    if (list1->val < list2->val) {
        list1->next = mergeTwoLists(list1->next, list2);
        return list1;
    } else {
        list2->next = mergeTwoLists(list1, list2->next);
        return list2;
    }
}
ListNode* createLinkedList(const vector<int>& values) {
    if (values.empty()) return nullptr;
    ListNode* head = new ListNode(values[0]);
    ListNode* current = head;
    for (size_t i = 1; i < values.size(); ++i) {
        current->next = new ListNode(values[i]);
        current = current->next;
    }
    return head;
}
void printLinkedList(ListNode* head) {
    while (head) {
        cout << head->val;
        if (head->next) cout << " -> ";
        head = head->next;
```



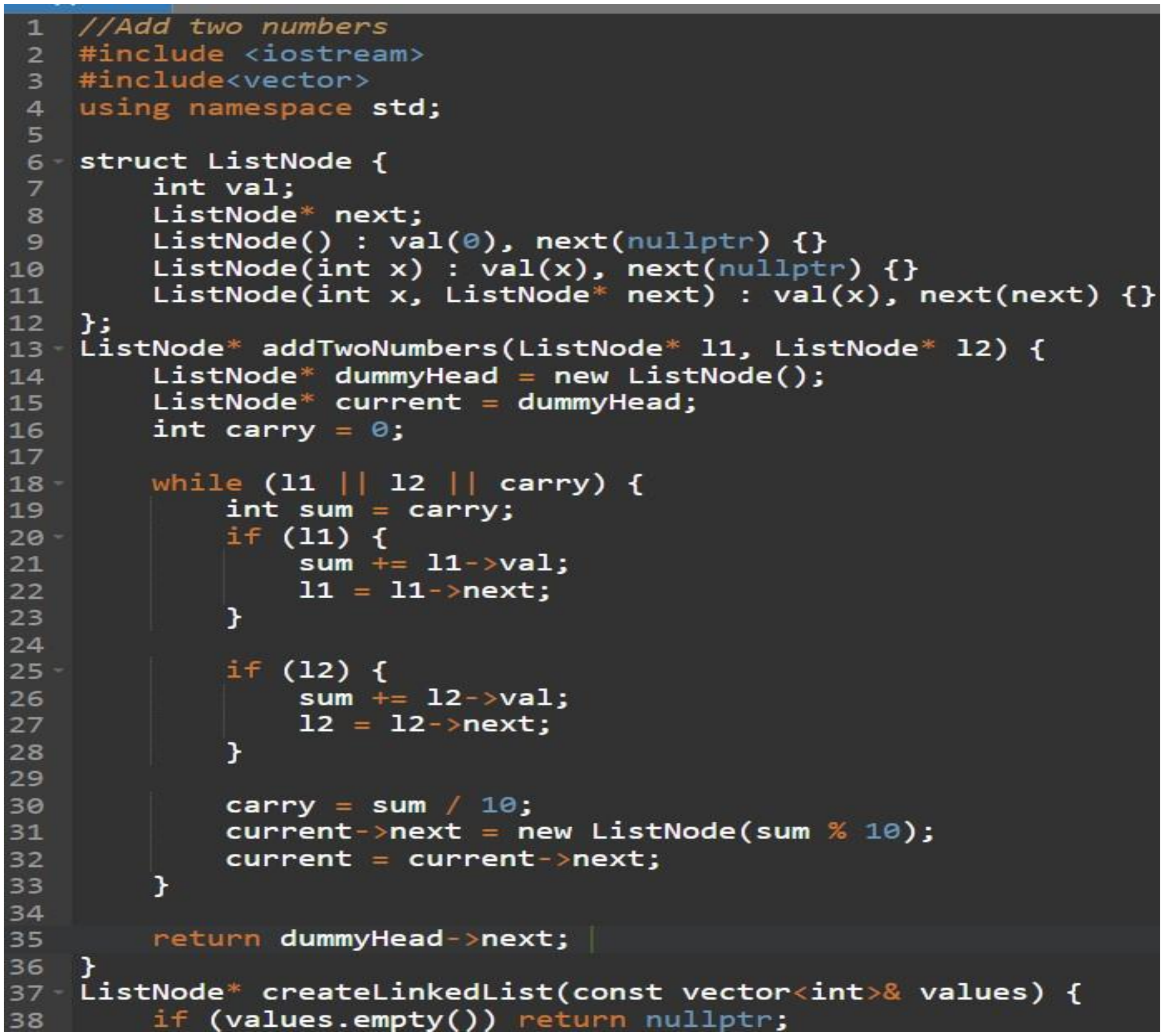
```cpp
    cout << endl;
}

int main() {
    vector<int> list1Values = {1, 2, 4};
    vector<int> list2Values = {1, 3, 4};
    ListNode* list1 = createLinkedList(list1Values);
    ListNode* list2 = createLinkedList(list2Values);
    ListNode* mergedList = mergeTwoLists(list1, list2);
    cout << "Merged List: ";
    printLinkedList(mergedList);

    return 0;
}
```

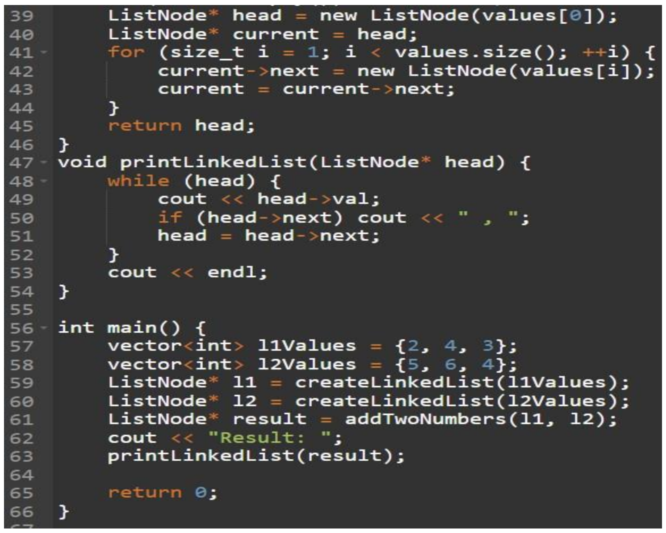
OUTPUT:


```
Merged List: 1 -> 1 -> 2 -> 3 -> 4 -> 4


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

Q3. Add two uumbers(medium).

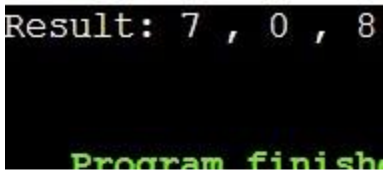
Sol.


```cpp
//Add two numbers
#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) {}
};
ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    ListNode* dummyHead = new ListNode();
    ListNode* current = dummyHead;
    int carry = 0;

    while (l1 || l2 || carry) {
        int sum = carry;
        if (l1) {
            sum += l1->val;
            l1 = l1->next;
        }

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

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

    return dummyHead->next;
}
ListNode* createLinkedList(const vector<int>& values) {
    if (values.empty()) return nullptr;
```



```
    ListNode* head = new ListNode(values[0]);
    ListNode* current = head;
    for (size_t i = 1; i < values.size(); ++i) {
        current->next = new ListNode(values[i]);
        current = current->next;
    }
    return head;
}
void printLinkedList(ListNode* head) {
    while (head) {
        cout << head->val;
        if (head->next) cout << " , ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    vector<int> l1Values = {2, 4, 3};
    vector<int> l2Values = {5, 6, 4};
    ListNode* l1 = createLinkedList(l1Values);
    ListNode* l2 = createLinkedList(l2Values);
    ListNode* result = addTwoNumbers(l1, l2);
    cout << "Result: ";
    printLinkedList(result);

    return 0;
}
```

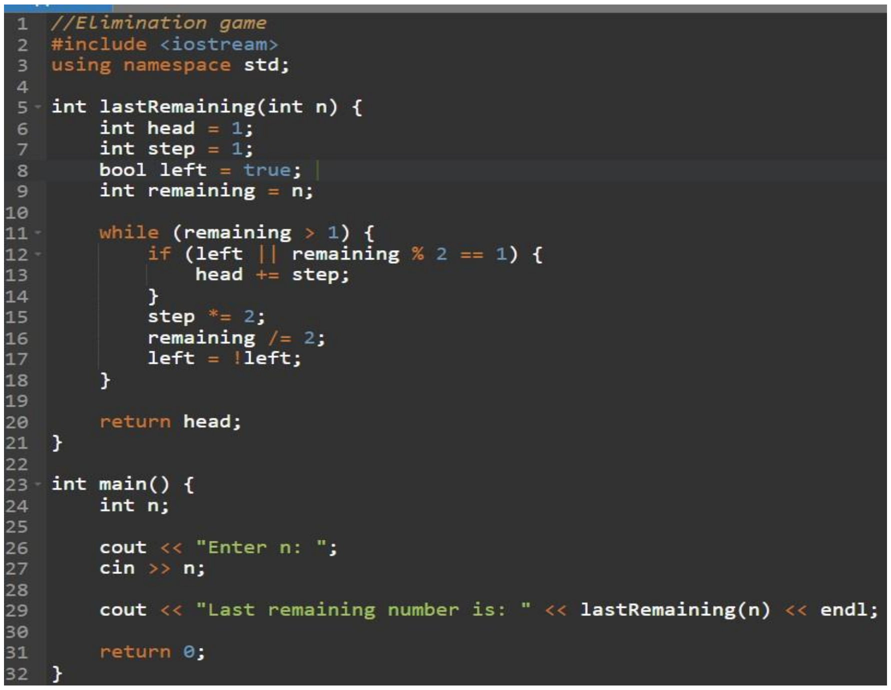
OUTPUT:



```
Result: 7 , 0 , 8


  Program finish
```

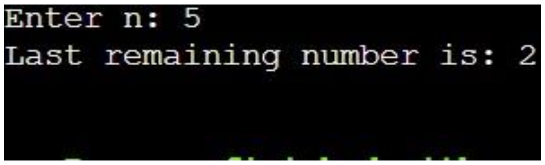
Q4.Elimination game hard)

Sol.



```
//Elimination game
#include <iostream>
using namespace std;

int lastRemaining(int n) {
    int head = 1;
    int step = 1;
    bool left = true;
    int remaining = n;

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

    return head;
}

int main() {
    int n;

    cout << "Enter n: ";
    cin >> n;

    cout << "Last remaining number is: " << lastRemaining(n) << endl;

    return 0;
}
```

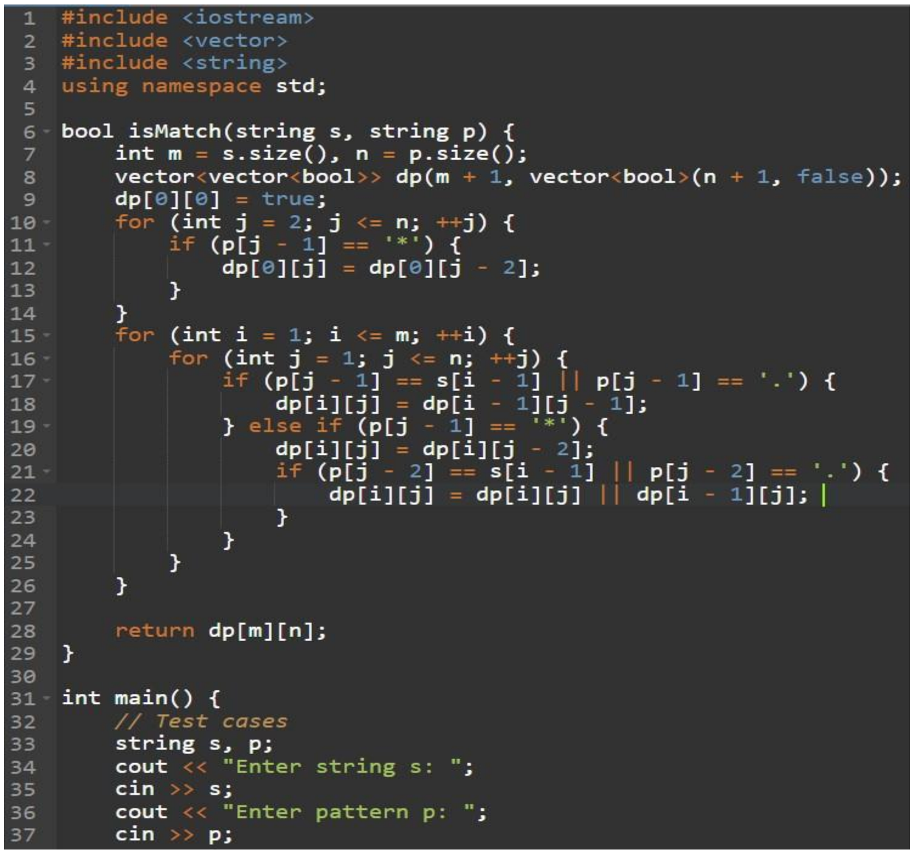
OUTPUT:



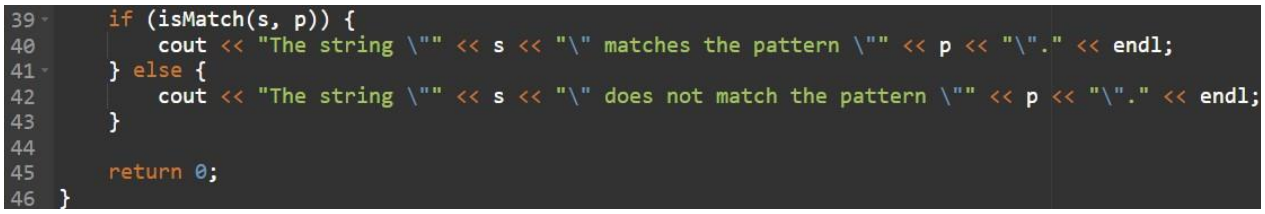
```
Enter n: 5
Last remaining number is: 2
```

Q5. Regular expression matching.(hard)

Sol.

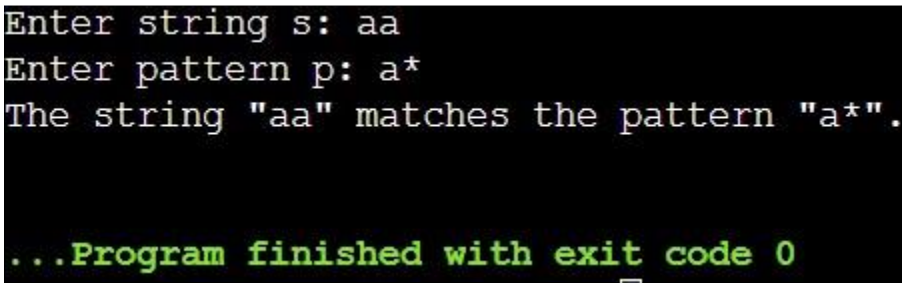


```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

bool isMatch(string s, string p) {
    int m = s.size(), n = p.size();
    vector<vector<bool>> dp(m + 1, vector<bool>(n + 1, false));
    dp[0][0] = true;
    for (int j = 2; j <= n; ++j) {
        if (p[j - 1] == '*') {
            dp[0][j] = dp[0][j - 2];
        }
    }
    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];
                if (p[j - 2] == s[i - 1] || p[j - 2] == '.') {
                    dp[i][j] = dp[i][j] || dp[i - 1][j];
                }
            }
        }
    }

    return dp[m][n];
}

int main() {
    // Test cases
    string s, p;
    cout << "Enter string s: ";
    cin >> s;
    cout << "Enter pattern p: ";
    cin >> p;
```



```cpp
    if (isMatch(s, p)) {
        cout << "The string \"" << s << "\" matches the pattern \"" << p << "\"." << endl;
    } else {
        cout << "The string \"" << s << "\" does not match the pattern \"" << p << "\"." << endl;
    }

    return 0;
}
```

OUTPUT:



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


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