Design and Analysis of Algorithms:

Lab Assessment -2

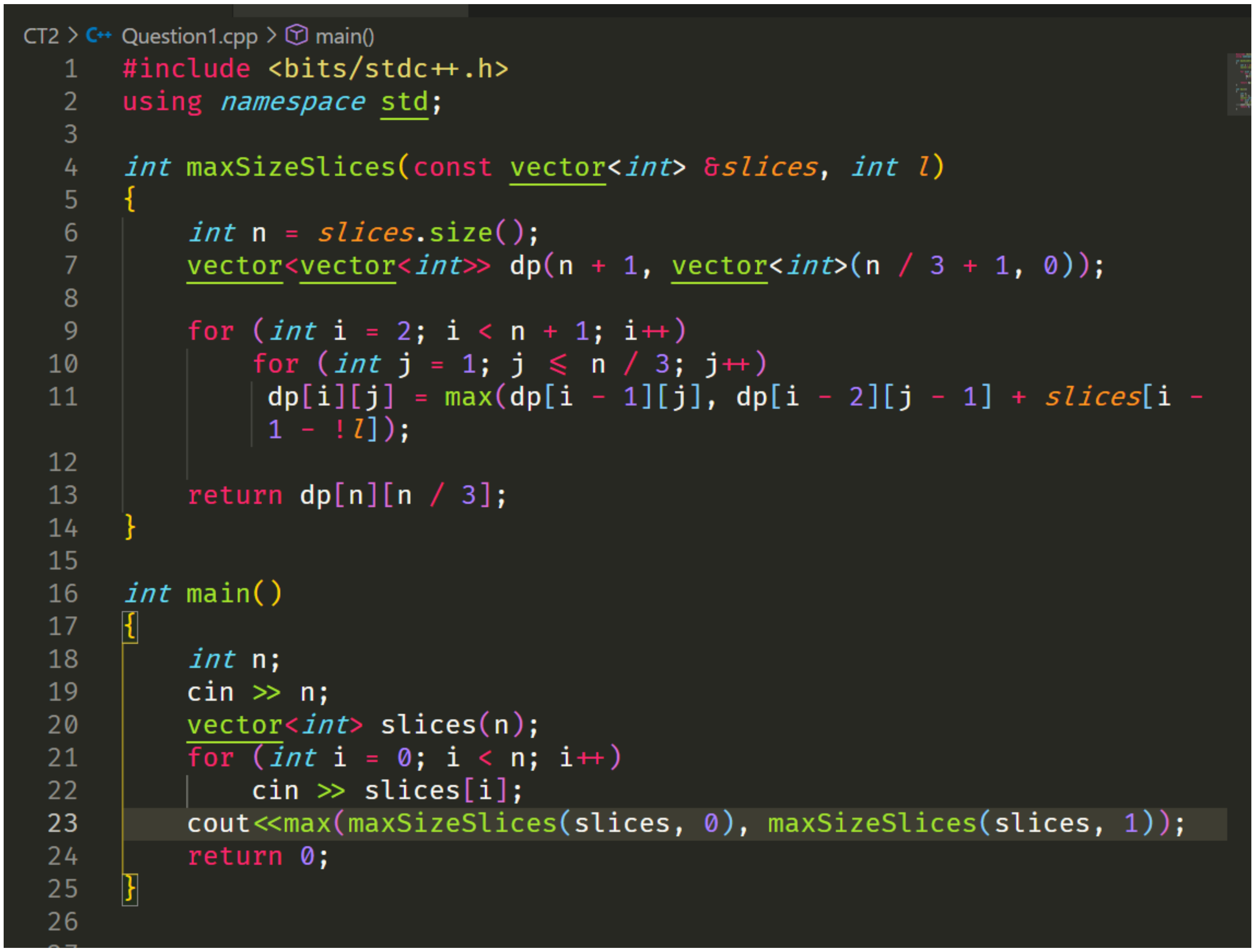
106119100 - Rajneesh Pandey

Question 1 :

Approach : First, observe that there exists no solution where we can pick both slice 0 and slice n - 1, which means that we can split the circular problem into 2 linear problems where we once pick slices from [0 ... n - 2] and then from [1 ... n - 1] (inclusive)

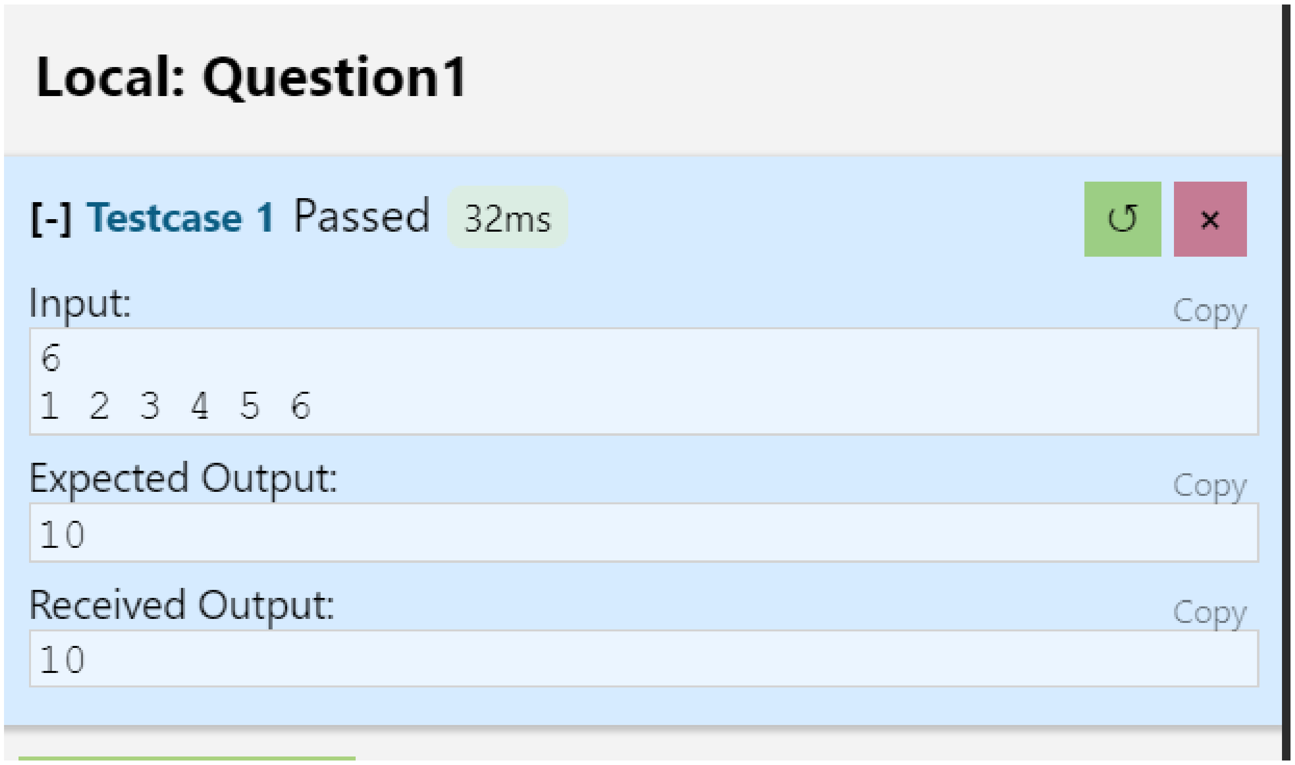
Basically, we want to choose the best way of selecting n / 3 slices such that no 2 slices are adjacent. Now for each slice, either we pick it or we do not. If we don't pick it, we are free to pick the one before it, else we must discard the one before it.

Hence, dp[i][j] is the best way to pick j slices among slices from [0 ... i - 2] or [1 ... i - 1] (depending on which of the two linear solutions we are currently handling, which is why you see slices[i - 1 - !l] in the solution)


CT2 > Question1.cpp > main()

```cpp
#include <bits/stdc++.h>
using namespace std;

int maxSizeSlices(const vector<int> &slices, int l)
{
    int n = slices.size();
    vector<vector<int>> dp(n + 1, vector<int>(n / 3 + 1, 0));

    for (int i = 2; i < n + 1; i++)
        for (int j = 1; j <= n / 3; j++)
         dp[i][j] = max(dp[i - 1][j], dp[i - 2][j - 1] + slices[i -
         1 - !l]);

    return dp[n][n / 3];
}

int main()
{
    int n;
    cin >> n;
    vector<int> slices(n);
    for (int i = 0; i < n; i++)
        cin >> slices[i];
    cout<<max(maxSizeSlices(slices, 0), maxSizeSlices(slices, 1));
    return 0;
}

```

Input / Output



Local: Question1

[-] Testcase 1 Passed 32ms

Input:

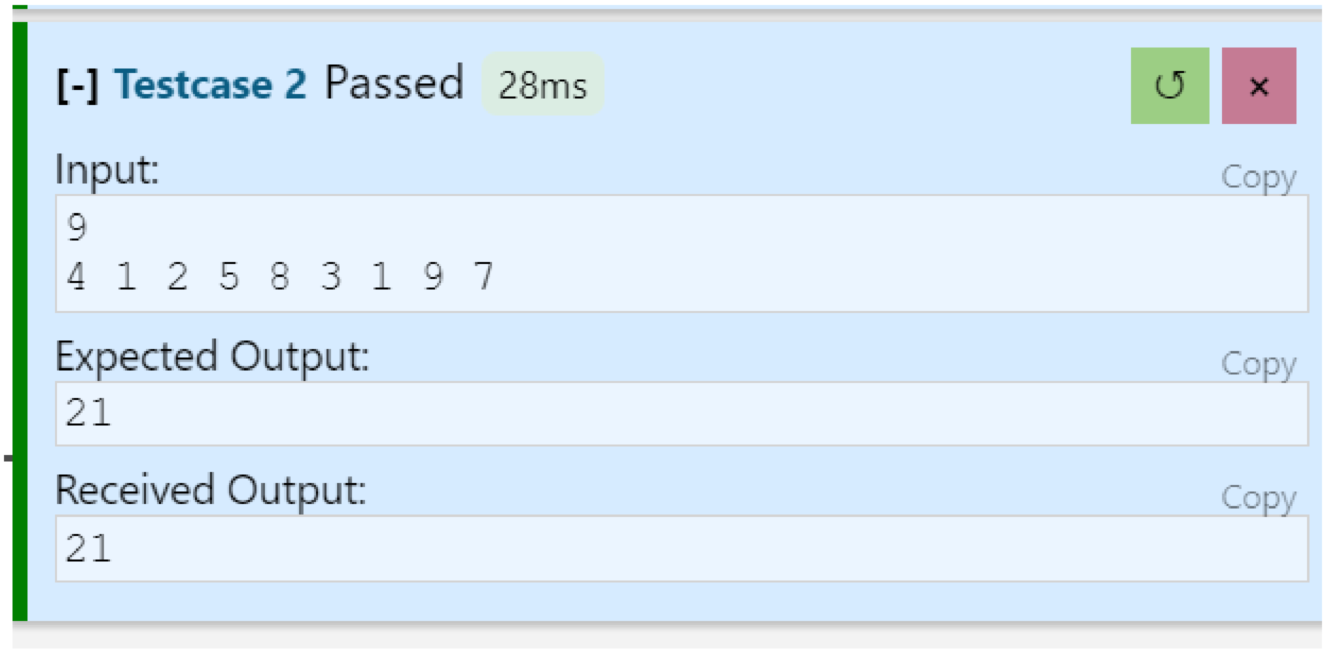
```
6
1 2 3 4 5 6
```

Expected Output:

```
10
```

Received Output:

```
10
```



[-] Testcase 2 Passed 28ms

Input:

```
9
4 1 2 5 8 3 1 9 7
```

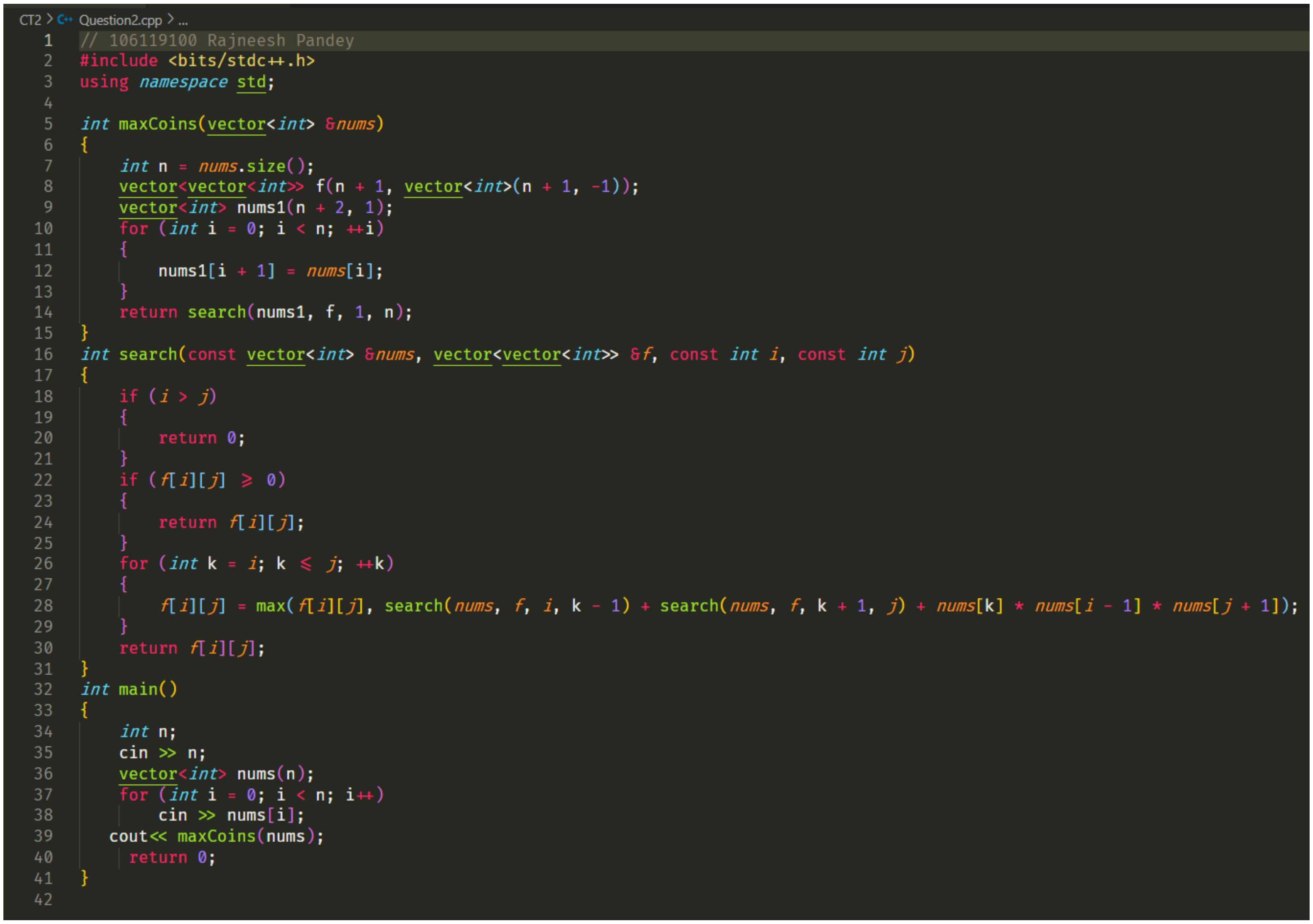
Expected Output:

```
21
```

Received Output:

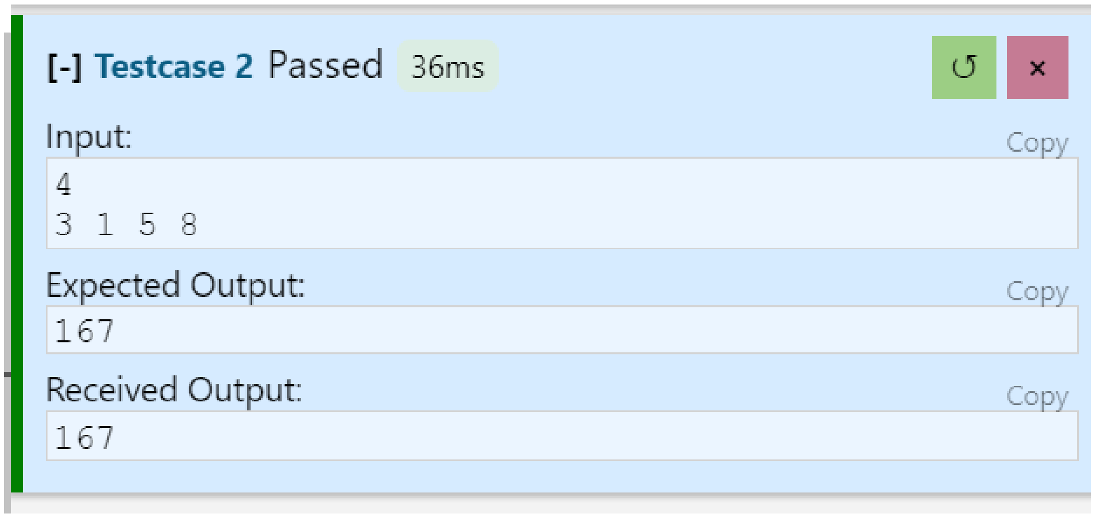
```
21
```

Question 2



CT2 > Question2.cpp > ...

```cpp
// 106119100 Rajneesh Pandey
#include <bits/stdc++.h>
using namespace std;

int maxCoins(vector<int> &nums)
{
    int n = nums.size();
    vector<vector<int>> f(n + 1, vector<int>(n + 1, -1));
    vector<int> nums1(n + 2, 1);
    for (int i = 0; i < n; ++i)
    {
        nums1[i + 1] = nums[i];
    }
    return search(nums1, f, 1, n);
}
int search(const vector<int> &nums, vector<vector<int>> &f, const int i, const int j)
{
    if (i > j)
    {
        return 0;
    }
    if (f[i][j] >= 0)
    {
        return f[i][j];
    }
    for (int k = i; k <= j; ++k)
    {
        f[i][j] = max(f[i][j], search(nums, f, i, k - 1) + search(nums, f, k + 1, j) + nums[k] * nums[i - 1] * nums[j + 1]);
    }
    return f[i][j];
}
int main()
{
    int n;
    cin >> n;
    vector<int> nums(n);
    for (int i = 0; i < n; i++)
        cin >> nums[i];
   cout<< maxCoins(nums);
     return 0;
}
```



[-] Testcase 2 Passed 36ms

Input:

```
4
3 1 5 8
```

Expected Output:

```
167
```

Received Output:

```
167
```