DAY-4 WINTER WINNING CAMP

NAME: Priti **Section: Kpit-901**

UID: 22BCS12729

Problem-1(Very Easy)

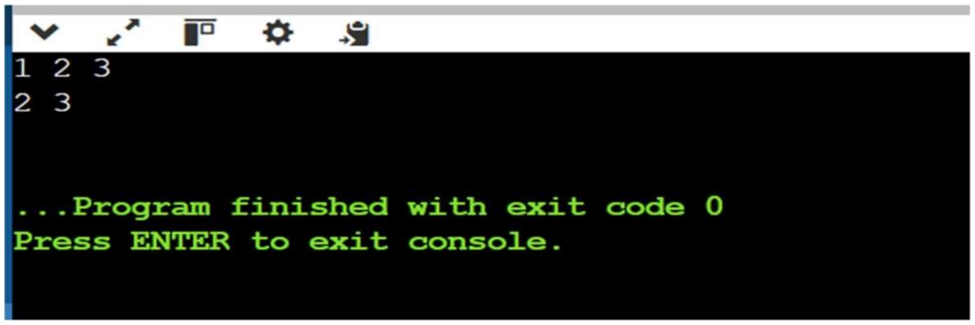
CODE:

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

class MyQueue {
private:
    stack<int> stack1;
    stack<int> stack2;

    void transferStack1ToStack2() {
        while (!stack1.empty()) {
            stack2.push(stack1.top());
            stack1.pop();
        }
    }

public:
    MyQueue() {}

    void push(int x) {
        stack1.push(x);
    }
```

```
int pop() {
    if (stack2.empty()) {
        transferStack1ToStack2();
    }
    int topElement = stack2.top();
    stack2.pop();
    return topElement;
}

int peek() {
    if (stack2.empty()) {
        transferStack1ToStack2();
    }
    return stack2.top();
}

bool empty() {
    return stack1.empty() && stack2.empty();
}

void printQueue() {
    if (stack2.empty()) {
        transferStack1ToStack2();
    }
    stack<int> temp = stack2;
    stack<int> tempStack1 = stack1;

    while (!temp.empty()) {
        cout << temp.top() << " ";
```

```
            temp.pop();
        }
        while (!tempStack1.empty()) {
            cout << tempStack1.top() << " ";
            tempStack1.pop();
        }
        cout << endl;
    }
};
int main() {
    MyQueue myQueue;
    myQueue.push(1);
    myQueue.push(2);
    myQueue.push(3);
    myQueue.printQueue();
    int front = myQueue.peek();
    int popped = myQueue.pop();
    myQueue.printQueue();
    bool isEmpty = myQueue.empty();
    return 0;
}
```

OUTPUT:



```
1 2 3
2 3


...Program finished with exit code 0
Press ENTER to exit console.
```

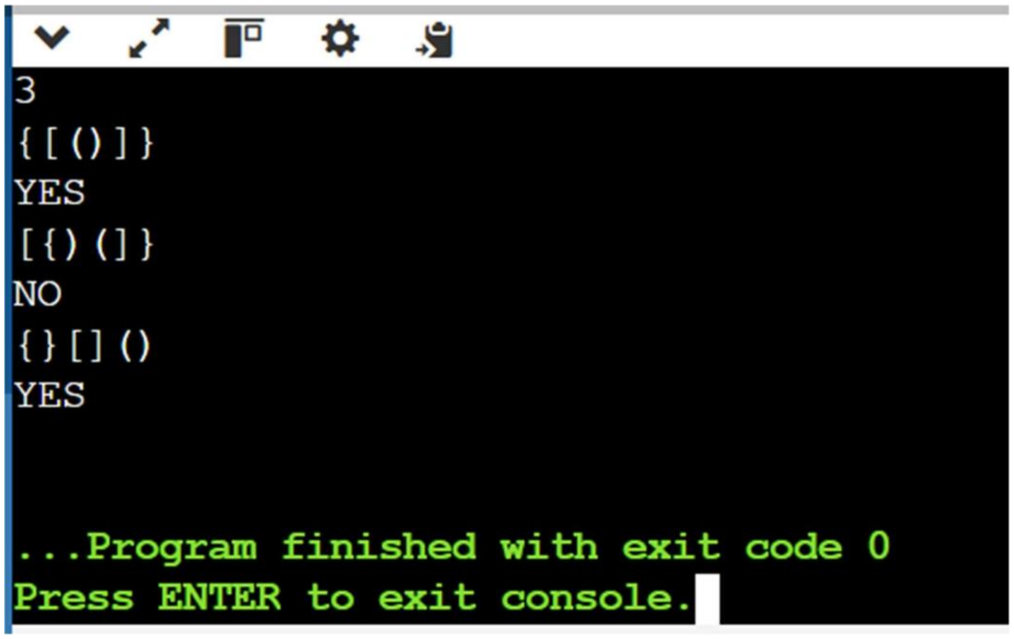
Problem-2(Easy)

CODE:

```
#include <iostream>

#include <stack>

#include <string>

using namespace std;


string isBalanced(string s) {

    stack<char> bracketStack;


    for (char c : s) {

        if (c == '(' || c == '{' || c == '[') {

            bracketStack.push(c);

        } else {

            if (bracketStack.empty()) {

                return "NO";

            }


            char top = bracketStack.top();

            if ((c == ')' && top == '(') || (c == '}' && top == '{') || (c == ']' && top == '[')) {

                bracketStack.pop();

            } else {

                return "NO";

            }

        }

    }


    return bracketStack.empty() ? "YES" : "NO";

}
```

```
int main() {
    int n;
    cin >> n;
    while (n--) {
        string s;
        cin >> s;
        cout << isBalanced(s) << endl;
    }
    return 0;
}
```

OUTPUT:



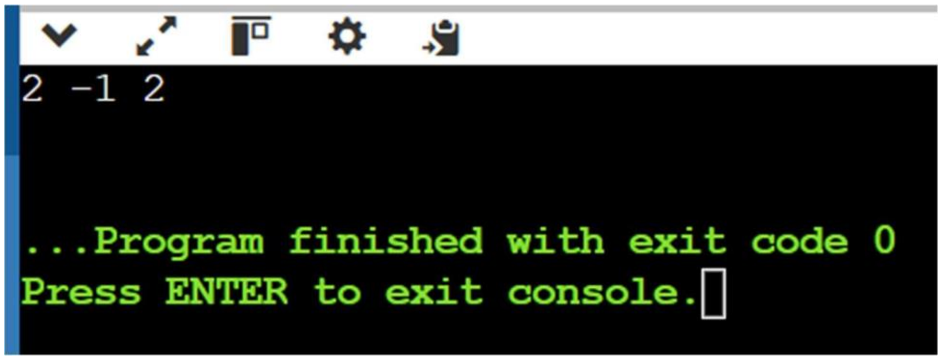
```
3
{[()]}
YES
[{)(]}
NO
{}[]()
YES


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem-3(Medium)

CODE:

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

vector<int> nextGreaterElements(vector<int>& nums) {
    int n = nums.size();
    vector<int> result(n, -1);
    stack<int> indexStack;

    for (int i = 0; i < 2 * n; ++i) {
        while (!indexStack.empty() && nums[indexStack.top()] < nums[i % n]) {
            result[indexStack.top()] = nums[i % n];
            indexStack.pop();
        }
        if (i < n) {
            indexStack.push(i);
        }
    }

    return result;
}

int main() {
    vector<int> nums = {1, 2, 1};
    vector<int> result = nextGreaterElements(nums);
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

OUTPUT:



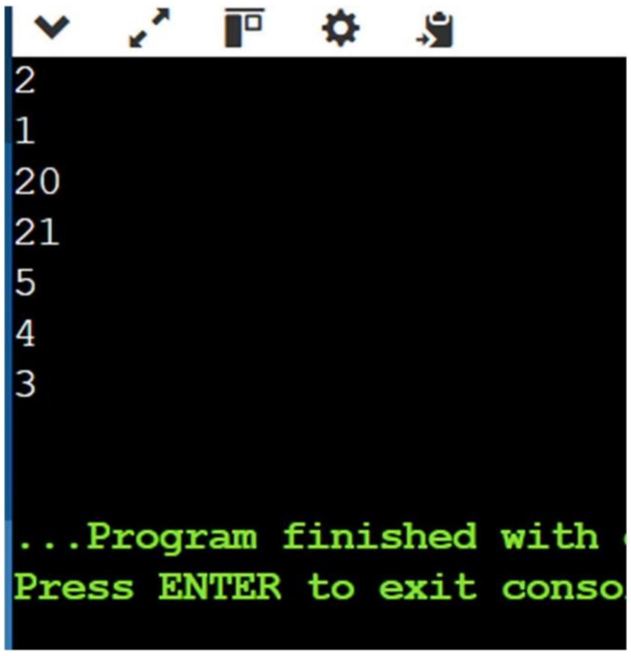
```
2 -1 2


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem-4(Hard)

CODE:

```
#include <iostream>

#include <vector>

#include <deque>

using namespace std;


class DinnerPlates {

private:

    vector<deque<int>> stacks;

    int capacity;

    deque<int> leftmostEmptyStack;

    int rightmostNonEmptyStack;

public:

    DinnerPlates(int capacity) : capacity(capacity), rightmostNonEmptyStack(-1) {}

    void push(int val) {

        if (!leftmostEmptyStack.empty()) {

            int index = leftmostEmptyStack.front();

            leftmostEmptyStack.pop_front();

            stacks[index].push_back(val);

            if (stacks[index].size() == capacity) {

                rightmostNonEmptyStack = max(rightmostNonEmptyStack, index);

            }

        } else {

            stacks.push_back({val});

            leftmostEmptyStack.push_back(stacks.size() - 1);

            rightmostNonEmptyStack = max(rightmostNonEmptyStack, 
static_cast<int>(stacks.size() - 1));

        }

    }
    int pop() {
        if (rightmostNonEmptyStack == -1) return -1;
        int val = stacks[rightmostNonEmptyStack].back();
        stacks[rightmostNonEmptyStack].pop_back();
        if (stacks[rightmostNonEmptyStack].empty()) {
            rightmostNonEmptyStack--;
        }
        if (stacks[rightmostNonEmptyStack].size() < capacity) {
            leftmostEmptyStack.push_back(rightmostNonEmptyStack);
        }
        return val;
    }
    int popAtStack(int index) {
        if (index < 0 || index >= stacks.size() || stacks[index].empty()) {
            return -1;
        }
        int val = stacks[index].back();
        stacks[index].pop_back();

        if (stacks[index].empty()) {
            leftmostEmptyStack.push_back(index);
        }
        return val;
    }
};
int main() {
    DinnerPlates dp(2);
    dp.push(1);
    dp.push(2);
    dp.push(3);

    dp.push(4);

    dp.push(5);

    cout << dp.popAtStack(0) << endl;

    dp.push(20);

    dp.push(21);

    cout << dp.popAtStack(0) << endl;

    cout << dp.popAtStack(2) << endl;

    cout << dp.pop() << endl;

    cout << dp.pop() << endl;

    cout << dp.pop() << endl;

    cout << dp.pop() << endl;


    return 0;

}
```

OUTPUT:



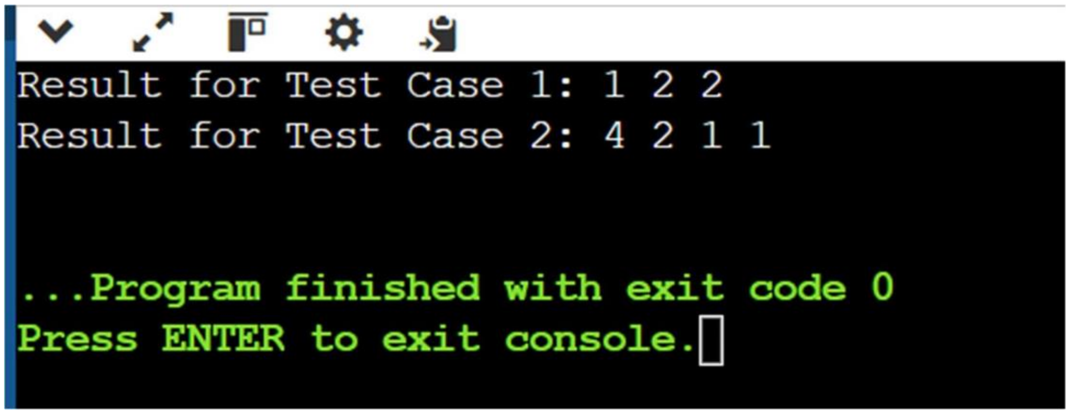
```
2
1
20
21
5
4
3


...Program finished with
Press ENTER to exit conso
```

Problem-5(VeryHard)

CODE:

```
#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>


 using namespace std;


vector<int> gcdPairs(vector<int>& nums, vector<int>& queries) {

   vector<int> gcdPairs;


    int n = nums.size();

   for (int i = 0; i < n; i++) {

      for (int j = i + 1; j < n; j++) {

         gcdPairs.push_back(gcd(nums[i], nums[j]));

      }

   }


   sort(gcdPairs.begin(), gcdPairs.end());


   vector<int> result;

   for (int query : queries) {

      result.push_back(gcdPairs[query]);

   }


   return result;

}


int main() {
    vector<int> nums1 = {2, 3, 4};
    vector<int> queries1 = {0, 2, 2};
    vector<int> result1 = gcdPairs(nums1, queries1);

    cout << "Result for Test Case 1: ";
    for (int val : result1) {
        cout << val << " ";
    }
    cout << endl;

    vector<int> nums2 = {4, 4, 2, 1};
    vector<int> queries2 = {5, 3, 1, 0};
    vector<int> result2 = gcdPairs(nums2, queries2);

    cout << "Result for Test Case 2: ";
    for (int val : result2) {
        cout << val << " ";
    }
    cout << endl;
    return 0;
}
```

OUTPUT:



```
Result for Test Case 1: 1 2 2
Result for Test Case 2: 4 2 1 1


...Program finished with exit code 0
Press ENTER to exit console.
```