MINI PROJECT

Project Title

POCKET CUBE

Guided by
Mr. S. J. Murchite

GROUP MEMBERS

Gidwani Manav Krupal	19UCS038
Jadhav Gaurav Nivas	19UCS045
Ekal Priyanshu Prakash	19UCS033
Kale Shubhankar Suhas	19UCS052

OUTLINE

- Introduction
- Problem Statement
- Problem Description
- Input and Output
- Requirement Analysis
- Algorithm
- Snapshot
- Conclusion
- References
- Any Question

INTRODUCTION

The mission of the Pocket Cube Project is to count the number of completed faces initially and after each rotation, either clockwise or counter clockwise. Our main goal is to develop this application to show that how a Rubik's 2x2x2 pocket cube is get solved by denoting the output as complete faces.

PROBLEM STATEMENT

To show the maximum number of completed faces in no more than N twist steps of a 2x2x2 Pocket Cube. Each small face of the cube (indexed from 0-23) forms a color which is denoted by a number. Such 6 numbers (each occurring 4 times) are present on the cube (indexed from 0 to 5).

PROBLEM DESCRIPTION

- Pocket Cube is a 2x2x2 cube, that is, it consists of 8 mini cubes.
- For a combination of 4 cubes (2x2 mini-cubes), sharing a whole cube face, you can twist it 90 degrees in clockwise or counter-clockwise direction.
- This twist operation is called a one twist step.
- There are total 24 mini-faces having 6 different colors (indexed from 0 to 5).
- If any 4 mini-cubes` faces are of same color relying on same large face of cube, we can call that face as completed face.
- Given you a color arrangement of all 24 faces from a scrambled Pocket Cube, output the maximum possible number of completed faces in no more than N twist steps.

INPUT

There will be several test cases. In each test case, there will be 2 lines. One integer *N* (1 <= N<=7) in the first line, then 24 integers Ci separated by a single space in the second line. For index 0 <=i<24, Ci is colour of the corresponding face.

Sample Input


```
1
0 0 0 0 1 1 2 2 3 3 1 1 2 2 3 3 4 4 4 4 5 5 5 5
1
0 4 0 4 1 1 2 5 3 3 1 1 2 5 3 3 4 0 4 0 5 2 5 2
```

OUTPUT

For each test case, output the maximum number of completed faces during no more than N twist steps.

Sample Output

6
2

REQUIREMENT ANALYSIS

For Case 1, the index 0,1,2,3 has 0,0,0,0 which is a complete face. Similarly, 4,5,10,11 has 1,1,1,1 ; 6,7,12,13 has 2,2,2,2 ; 8,9,14,15 has 3,3,3,3 ; 16,17,18,19 has 4,4,4,4 ; 20,21,22,23 has 5,5,5,5, which are the complete faces.

index :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
values:	0	0	0	0	1	1	2	2	3	3	1	1	2	2	3	3	4	4	4	4	5	5	5	5

Also, for Case 2, the index 4,5,10,11 having 1,1,1,1 & 8,9,14,15 having 3,3,3,3 is complete faces while remaining are flipped vertically top ward as shown below:

index :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
values:	0	4	0	4	1	1	2	5	3	3	1	1	2	5	3	3	4	0	4	0	5	2	5	2

REQUIREMENT ANALYSIS

For Case 3, the index 6,7,12,13 having 2,2,2,2 & 20,21,22,23 having 5,5,5,5 is complete faces while remaining are flipped horizontally in anticlockwise direction as shown below:

index : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
values: 4 4 0 0 3 1 2 2 3 1 3 1 2 2 3 1 4 4 0 0 5 5 5 5

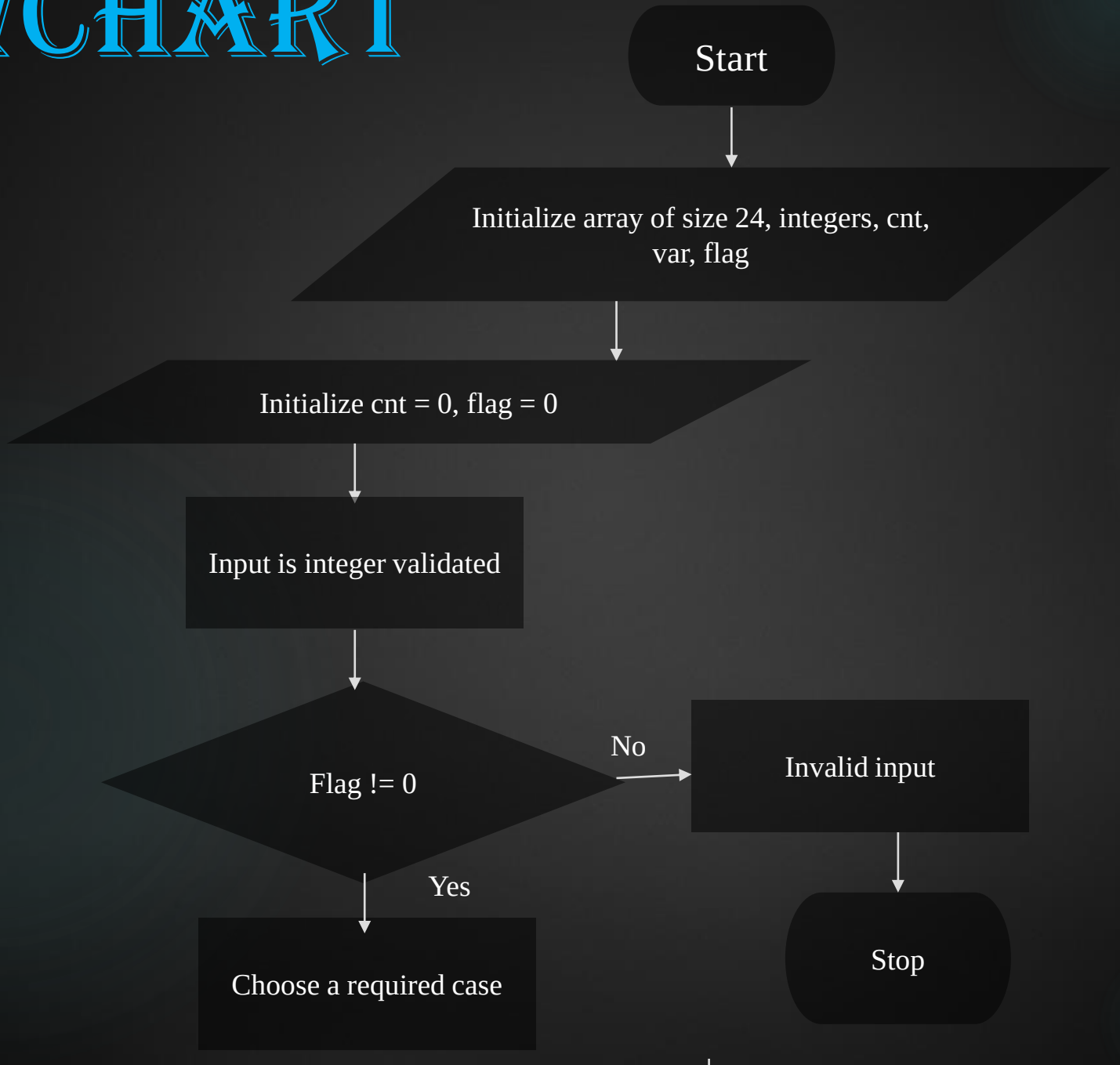
For Case 4, the index 6,7,12,13 having 2,2,2,2 & 20,21,22,23 having 5,5,5,5 is complete faces while remaining are flipped horizontally in clockwise direction
as shown below:

index : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
values: 0 0 4 4 1 3 2 2 1 3 1 3 2 2 1 3 0 0 4 4 5 5 5 5

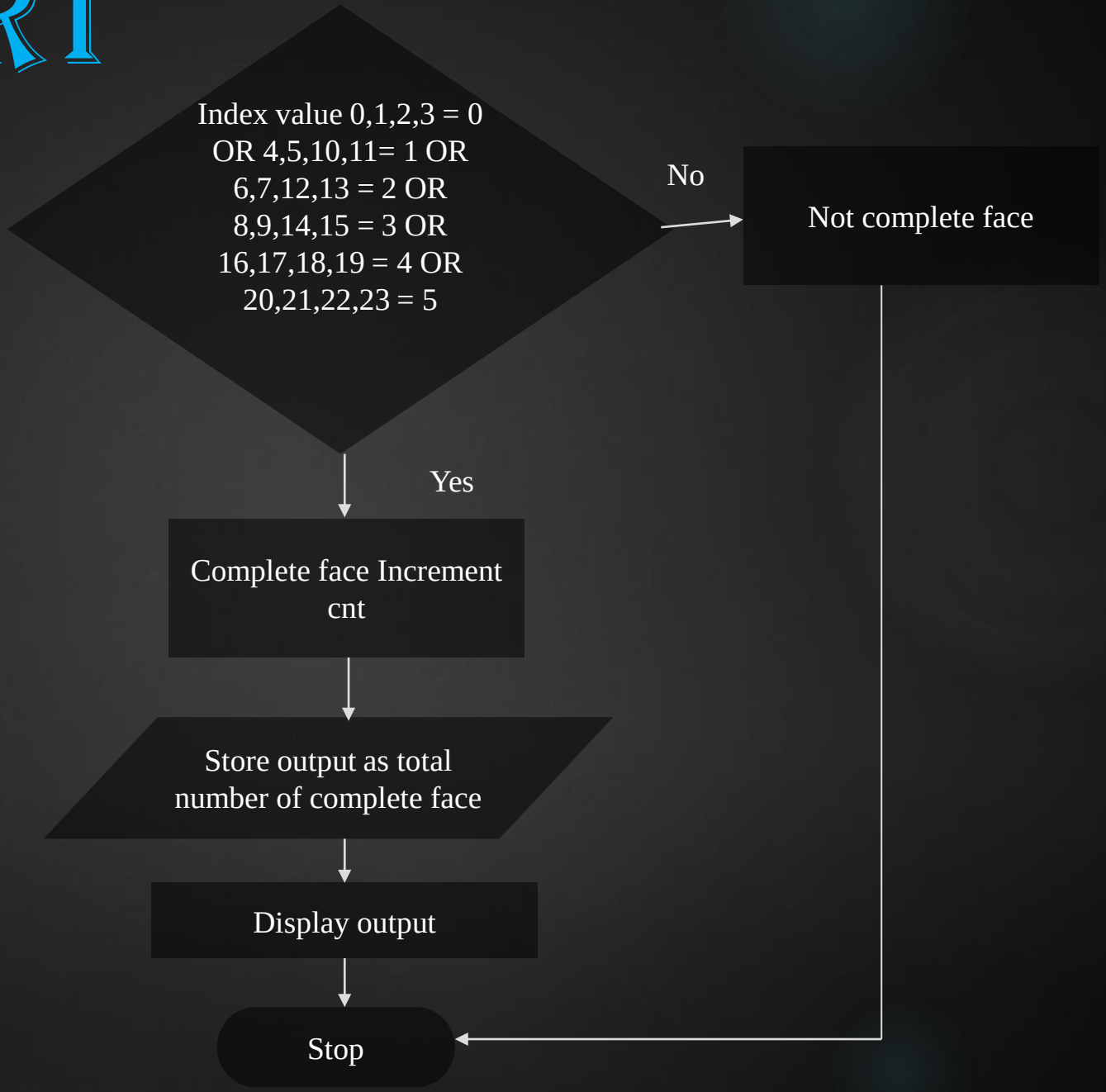
ALGORITHM

1. Start
2. Initialize an array C of size 24 as first case, integers i, cnt, var.
3. Initialize cnt variable to 0.
4. Similar to array C, initialize array B, D, E of size 24 as second, third, fourth case respectively.
5. Ask user to choose the required case.
6. Using for loop initializing i from 0 to length of variable var, if value of var is false then make the flag status equal to 0 & terminate the program.
7. If flag is not equal to 0, then print that input is integer validated.
8. Store the case as integer in var variable.
9. Using switch-case and if-else condition, check out whether the face is complete or not.
10. If complete, increment the cnt variable by 1.
11. Output the variable cnt as number of complete faces.
12. If user inputs a different case, pop-up a message as invalid input.
13. End.

FLOWCHART

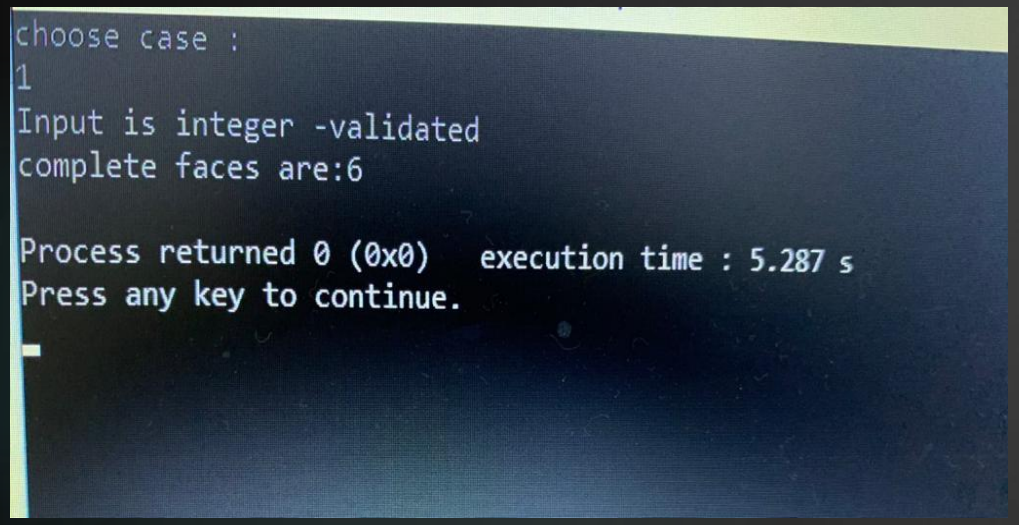

Start
Initialize array of size 24, integers, cnt, var, flag
Initialize cnt = 0, flag = 0
Input is integer validated
Flag != 0
No
Invalid input
Stop
Yes
Choose a required case

FLOWCHART

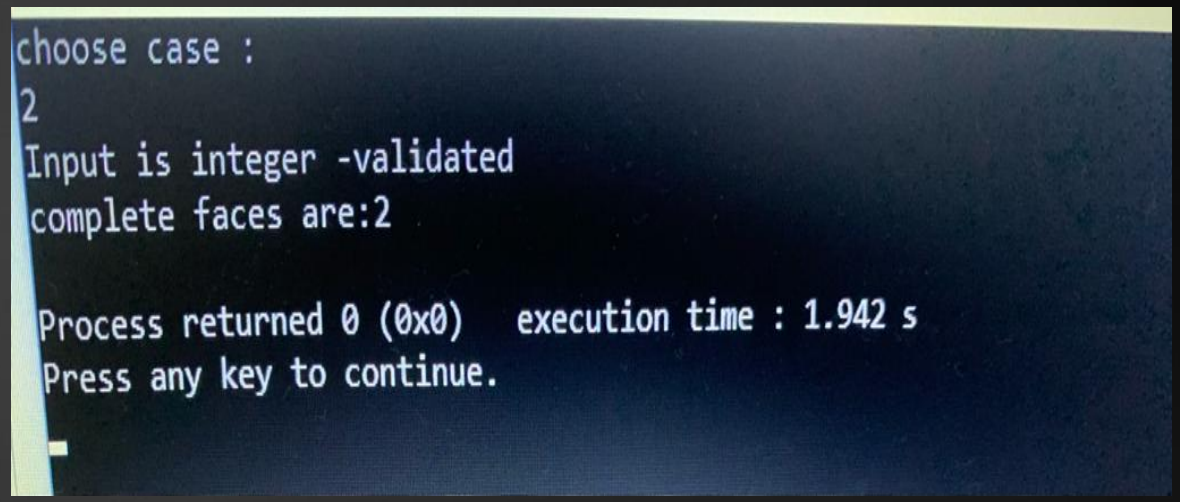

Index value 0,1,2,3 = 0
OR 4,5,10,11= 1 OR
6,7,12,13 = 2 OR
8,9,14,15 = 3 OR
16,17,18,19 = 4 OR
20,21,22,23 = 5
No
Not complete face
Yes
Complete face Increment cnt
Store output as total number of complete face
Display output
Stop

SNAPSHOT

Case 1:



Case 2:



Case 3:

```
choose case :
3
Input is integer -validated
complete faces are:2

Process returned 0 (0x0)   execution time : 8.215 s
Press any key to continue.
```

Case 4:

```
choose case :
4
Input is integer -validated
complete faces are:2

Process returned 0 (0x0)   execution time : 2.813 s
Press any key to continue.
```

Case 5.1:

```
choose case :
a
Invalid input program aborted
Process returned 0 (0x0)   execution time : 2.832 s
Press any key to continue.
```

Case 5.2:

```
choose case :
5
Input is integer -validated
invalid case

Process returned 0 (0x0)   execution time : 3.234 s
Press any key to continue.
```

CONCLUSION

As an overall conclusion we have achieved the purpose of the project to find out the number of complete faces of the cube based on the given test cases after the N twist steps.

REFERENCES

- Books : Let Us C++ - Yashwant Kanetkar
- Web-links :

  https://ruwix.com/twisty-puzzles/2x2x2-rubiks-cube-pocket/

ANY QUESTION ?

THANK YOU!