

WEEK 04

1. Let N be the total number of stones. In each turn, a player can remove either one stone or four stones. The player who picks the last stone, wins. They follow the "Ladies First" norm. Hence Alice is always the one to make the first move. Your task is to find out whether Alice can win, if both play the game optimally.

Input Format:

The first line starts with T , which is the number of test cases. Each test case will contain N number of stones.

Output Format:

Print "Yes" in the case Alice wins, else print "No".

Program:

```
1 #include<stdio.h>
2 int main(){
3     int t;
4     scanf("%d", &t);
5     for (int i = 0; i<t; i++){
6         int n;
7         scanf("%d", &n);
8         while (n > 0){
9             if (n >= 4){
10                 n -= 4;
11             }
12             else if (n >= 1){
13                 n -= 1;
14             }
15             if (n == 0){
16                 printf("Yes\n");
17                 break;
18             }
19             if (n >= 4){
20                 n -= 4;
21             }
22             else if (n >= 1){
23                 n -= 1;
24             }
25             if (n == 0){
26                 printf("No\n");
27                 break;
28             }
29         }
30         return 0;
31     }
32 }
```

Output:

	Input	Expected	Got	
✓	3	Yes	Yes	✓
	1	Yes	Yes	
	6	No	No	
	7			

Passed all tests! ✓

2. The number of holes that each of the digits from 0 to 9 have are equal to the number of closed paths in the digit. Their values are:

1, 2, 3, 5, 7 = 0 holes.

0, 4, 6, 9 = 1 hole.

8 = 2 holes.

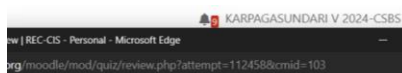
Given a number, you must determine the sum of the number of holes for all of its digits.

For example, the number 819 has 3 holes. Complete the program, it must return an integer denoting the total number of holes in num.

Input Format for Custom Testing:

There is one line of text containing a single integer num, the value to process.

Program:



```
1 #include<stdio.h>
2 int main(){
3     int n, count = 0;
4     scanf("%d", &n);
5     while (n > 0){
6         switch (n % 10){
7             case 0:
8             case 4:
9             case 6:
10            case 9:
11                count += 1;
12                break;
13            case 8:
14                count += 2;
15                break;
16        }
17        n = n/10;
18    }
19    printf("%d", count );
20    return 0;
21 }
```

Output:

	Input	Expected	Got	
✓	630	2	2	✓
✓	1288	4	4	✓
Passed all tests! ✓				

3. The problem solvers have found a new Island for coding and named it Philaland. Manish has come up with a solution that if we make coins category starting from \$1 till the maximum price of the item present on Island, then we can purchase any item easily. He added the following example to prove his point. Let's suppose the maximum price of an item is 5\$ then we can make coins of {\$1, \$2, \$3, \$4, \$5} to purchase any item ranging from \$1 till \$5. Now Manisha, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {\$1, \$2, \$3}. According to him Any item can be purchased at one time ranging from \$1 to \$5. Everyone was impressed with both of them. Your task is to help Manisha come up with a minimum number of denominations for any arbitrary max price in Philaland.

Input Format:

Contains an integer N denoting the maximum price of the item present on Philaland.

Output Format:

Print a single line denoting the minimum number of denominations of coins required.

Program:

```
KARPAGASUNDARI V 2024-CSBS
new | REC-CIS - Personal - Microsoft Edge
xg/moodle/mod/quiz/review.php?attempt=112458&cmid=103
```

```
1 #include<stdio.h>
2 int main(){
3     int n, i = 0;
4     scanf("%d", &n);
5     while (n>=10){
6         i +=1;
7         n = n/10;
8     }
9     if (n==5){
10        printf("%d", 3*i + 3);
11    }
12    else if (n==2){
13        printf("%d", 3*i + 2);
14    }
15    else if (n==1){
16        printf("%d", 3*i + 1);
17    }
18    return 0;
19 }
20
```

Output:

	Input	Expected	Got	
✓	10	4	4	✓
✓	5	3	3	✓
✓	20	5	5	✓
✓	500	9	9	✓
✓	1000	10	10	✓

Passed all tests! ✓

4. A set of N numbers (separated by one space) is passed as input to the program. The program must identify the count of numbers where the number is odd number.

Input Format:

The first line will contain the N numbers separated by one space.

Output Format:

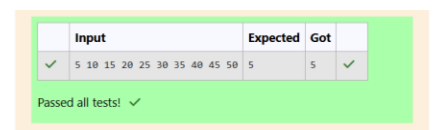
The count of numbers where the numbers are odd numbers.

Program:

A screenshot of a web browser showing a C program in a code editor. The browser's address bar shows 'z/review.php?attempt=1125148&cmid=104'. The code is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int n, count = 0;
5     char c;
6     do{
7         scanf("%d%c", &n, &c);
8         if (n % 2 != 0){
9             count += 1;
10        }
11    }while (c != '\n');
12    printf("%d", count);
13    return 0;
14 }
15 }
```

Output:

A screenshot of a test results table with a green background. The table has four columns: 'Input', 'Expected', 'Got', and a status column. The first row shows the input '5 10 15 20 25 30 35 40 45 50', an expected output of '5', and a 'Got' value of '5' with a checkmark. Below the table, it says 'Passed all tests!' with a checkmark.

Input	Expected	Got	
✓ 5 10 15 20 25 30 35 40 45 50	5	5	✓

Passed all tests! ✓

5. Given a number N, return true if and only if it is a confusing number, which satisfies the following condition:

We can rotate digits by 180 degrees to form new digits. When 0, 1, 6, 8, 9 are rotated 180 degrees, they become 0, 1, 9, 8, 6 respectively. When 2, 3, 4, 5 and 7 are rotated 180 degrees, they become invalid. A confusing number is a number that when rotated 180 degrees becomes a different number with each digit valid.

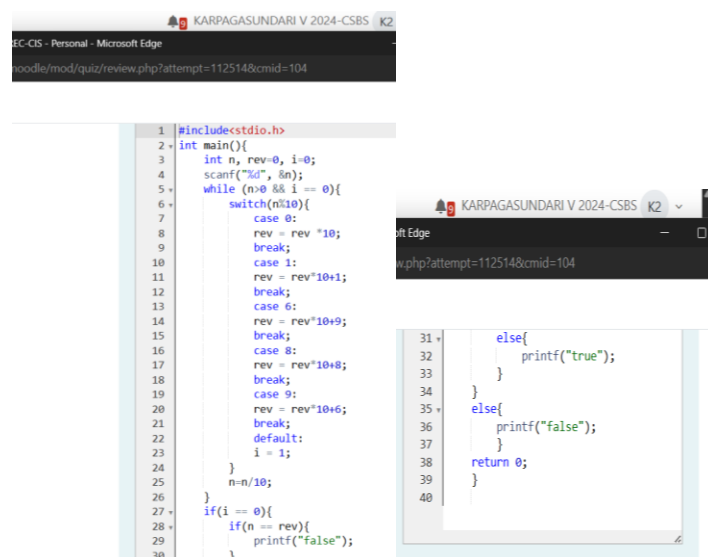
Example 1:

Input: 6

Output: true

Explanation: We get 9 after rotating 6, 9 is a valid number and $9 \neq 6$.

Program:



```
1 #include<stdio.h>
2 int main(){
3     int n, rev=0, i=0;
4     scanf("%d", &n);
5     while (n>0 && i == 0){
6         switch(n%10){
7             case 0:
8                 rev = rev * 10;
9                 break;
10             case 1:
11                 rev = rev * 10 + 1;
12                 break;
13             case 6:
14                 rev = rev * 10 + 9;
15                 break;
16             case 8:
17                 rev = rev * 10 + 8;
18                 break;
19             case 9:
20                 rev = rev * 10 + 6;
21                 break;
22             default:
23                 i = 1;
24         }
25         n = n / 10;
26     }
27     if(i == 0){
28         if(n == rev){
29             printf("false");
30         }
31     }
32     else{
33         printf("true");
34     }
35 }
36 else{
37     printf("false");
38 }
39 return 0;
40 }
```

Output: true

Output:

	Input	Expected	Got	
✓	6	true	true	✓
✓	89	true	true	✓
✓	25	false	false	✓
Passed all tests! ✓				

6. Every food item arranged in a single line will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food

item with value 2 has 2 macronutrients and incrementing in this fashion. The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value.

Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Input Format for Custom Testing

The first line contains an integer, n , that denotes the number of food items. The second line contains an integer, k , that denotes the unhealthy number.

Problem:

```
1 #include<stdio.h>
2 int main(){
3     int n,k,t = 0, c=1;
4     scanf("%d%d", &n, &k);
5     for (int i = 1; i<=n; i++){
6         t= (t+i)% 1000000007;
7         if( t==k) {
8             t -= k;
9             c++;
10        }
11    }
12    printf("%d",t);
13    return 0;
14 }
15 }
```

Output:

	Input	Expected	Got	
✓	2 2	3	3	✓
✓	2 1	2	2	✓
✓	3 3	5	5	✓

Passed all tests! ✓

