Infosys_Power_Programmer_Test_9

Test Summary

- No. of Sections: 1
- No. of Questions: 3
- Total Duration: 120 min

Section 1 - Coding

Section Summary

No. of Questions: 3
Duration: 120 min

Additional Instructions:

None

Q1. Abinesh has two Integers A and B, and he was to print all the terms of the series upto B-terms of the A-bonacci Numbers.
For example, when A = 2, the sequence becomes Fibonacci, when A = 3, sequence becomes Tribonacci.
Input : A = 3, B = 8
Output : 0, 0, 1, 1, 2, 4, 7, 13
We need to print first B terms.
First three terms are 0, 0 and 1.
Fourth term is 0 + 0 + 1 = 1
Fifth term is 0 + 1 + 1 = 2
Sixth terms is 1 + 1 + 2 = 4
Seventh term is 7 (1 + 2 + 4) and eighth
term is 13 (7 + 4 + 2).
Note:
If A=3, the series start with 0,0,1
If A=5, the series start with 0,0,0,0,1
If A=6, the series start with 0,0,0,0,0,1

Input Format

Enter the input number(A)
Enter the input number(B)

Output Format

Display the sequence which was obtained with the help of A and B.

Constraints

0 < A,B < 100
(Only positive integers are allowed)

Sample Input	Sample Output
3 8	0 0 1 1 2 4 7 13

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. Ganesh is a maths teacher, he gives a problem statement to his/her students to solve it. The question is, a number can always be represented as a sum of squares of other numbers. Note that 1 is a square and we can always break a number as (1*1 + 1*1 + 1*1 + …). Given a number n, the task is to find the minimum number of squares that sum to n.
For Ex:
Input: n = 100
Output: 1
100 can be written as 10^2. Note that 100 can also be written as $5^2 + 5^2 + 5^2 + 5^2$, but this representation requires 4 squares. So,the output is 1 since we have single square of 10 to satisfied the condition(minimum

number of squares that sum up to n.).

Input Format

Enter the input number(n)

Output Format

Display the minimum number of squares that sum to n.

Constraints

0 < n < 1000
(n should be positive integer, decimals are not allowed)

Sample Input	Sample Output
100	1

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. Anitha and Aarthi are sisters, they are playing a game in which Aarthi will provide a number. Anitha has to find out the number is divisible by 8 even after removing certain digits.
(Note:It's not allowed to rearrange the digits.)
For Ex:
Input : 1787075866
Output : Yes
There exist more one or more subsequences divisible by 8. Example subsequences are 176, 16 and 8.

Input Format

Enter the input number

Output Format

Display the Yes if the number is divisible by 8, even after removing certain digits or if it not divisible then display No.

Constraints

0 < len(Num) < 100
The length of the number should not exceed 100.

Sample Input	Sample Output
653468922	653468922 Yes

Answer Key & Solution

Section 1 - Coding

Q1 **Test Case**

Input	Output
10 19	0 0 0 0 0 0 0 0 0 1 1 2 4 8 16 32 64 1

Weightage - 10

Input

```
20
34
```

Output

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Weightage - 15

Input

```
8
10
```

Output

```
0 0 0 0 0 0 0 1 1 2
```

Weightage - 10

Input

```
9
13
```

Output

```
0 0 0 0 0 0 0 0 1 1 2 4 8
```

Weightage - 10

Input

```
23
30
```

Output

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Weightage - 15

Input

```
30
60
```

Output

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Weightage - 20

Input

```
34
65
```

Output

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Weightage - 20

Sample Input

```
3
8
```

Sample Output

```
0 0 1 1 2 4 7 13
```

Solution

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

```
// Function to print bonacci(n-acci) series
void bonacciseries(long n, int m)
{

    // Assuming m > n.
    int a[m] = { 0 };
    a[n - 1] = 1;
    a[n] = 1;
    for (int i = n + 1; i < m; i++)
        a[i] = 2 * a[i - 1] - a[i - n - 1];

    // displaying the results
    for (int i = 0; i < m; i++)
        cout << a[i] << " ";
}

//main function for displaying the series
int main()
{
    int A,B;
    cin>>A;
    cin>>B;
    bonacciseries(A,B);
    return 0;
}
```

Q2

Test Case

Input	Output
25	1

Weightage - 10

Input	Output
50	2

Weightage - 10

Input	Output
79	4

Weightage - 10

Input	Output
810	2

Weightage - 20

Input	Output
999	4

Weightage - 20

Input	Output
450	2

Weightage - 15

Input	Output
399	4

Weightage - 15

Sample Input	Sample Output
100	1

Solution

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

// Returns count of minimum squares that sum to n
int getMinSquares(int n)
{
    int* dp = new int[n + 1];

    // getMinSquares table for base case entries
    dp[0] = 0;
    dp[1] = 1;
    dp[2] = 2;
    dp[3] = 3;

    // getMinSquares rest of the table using recursive formula
    for (int i = 4; i <= n; i++) {
        // max value is i as i can always be represented as 1*1 + 1*1 + ...
        dp[i] = i;

        // going through all smaller numbers to recursively find minimum
        for (int x = 1; x <= ceil(sqrt(i)); x++) {
            int temp = x * x;
            if (temp > i)
                break;
            else
                dp[i] = min(dp[i], 1 + dp[i - temp]);
        }
    }
```

```
    // Storing the result and deleting the occupied spaces
    int res = dp[n];
    delete[] dp;

    return res;
}

// main program
int main()
{
    int n;
    cin>>n;
    cout << getMinSquares(n);
    return 0;
}
```

Q3

Test Case

Input	Output
100233002	100233002 Yes

Weightage - 10

Input	Output
277177011030	277177011030 Yes

Weightage - 10

Input	Output
77199	77199 No

Weightage - 10

Input	Output
99422313	99422313 Yes

Weightage - 10

Input	Output
758593780053	758593780053 Yes

Weightage - 10

Input	Output

93804020440424202929	93804020440424202929 Yes

Weightage - 20

Input	Output
910320237047027772471	910320237047027772471 Yes

Weightage - 20

Input	Output
15951579	15951579 No

Weightage - 10

Sample Input	Sample Output
653468922	653468922 Yes

Solution

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

bool isSubSeqDivisible(string str)
{
    int n = str.length();
    int dp[n + 1][10];
    memset(dp, 0, sizeof(dp));

    int arr[n + 1];
    for (int i = 1; i <= n; i++)
        arr[i] = str[i - 1] - '0';

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

        dp[i][arr[i] % 8] = 1;
        for (int j = 0; j < 8; j++) {

            if (dp[i - 1][j] > dp[i][(j * 10 + arr[i]) % 8])
                dp[i][(j * 10 + arr[i]) % 8] = dp[i - 1][j];

            if (dp[i - 1][j] > dp[i][j])
                dp[i][j] = dp[i - 1][j];
        }
    }

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

        if (dp[i][0] == 1)
            return true;
    }
```

```
    return false;
}

// main function
int main()
{
    string str;
    cin>>str;
    if (isSubSeqDivisible(str))
        cout << str << " Yes";
    else
        cout << str << " No";
    return 0;
}
```