Full Name:	Muhammad Nabil
Email:	cucibaju123@gmail.com
Test Name:	**Mock Test**
Taken On:	9 May 2024 09:02:49 IST
Time Taken:	0 min 45 sec/ 20 min
Invited by:	Ankush
Invited on:	9 May 2024 09:02:37 IST
Skills Score:	
Tags Score:	Algorithms 120/120 Core CS 120/120 Dynamic Programming 120/120 Medium 120/120 problem-solving 120/120

100%

120/120

scored in **Mock Test** in 0 min 45 sec on 9 May 2024 09:02:49 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Fibonacci Modified > **Coding**	35 sec	120/ 120	✓

QUESTION 1

Correct Answer

Score 120

Fibonacci Modified > Coding

Dynamic Programming | Algorithms | Medium | problem-solving | Core CS

QUESTION DESCRIPTION

Implement a *modified* Fibonacci sequence using the following definition:

Given terms $t[i]$ *and* $t[i+1]$ *where* $i \in (1, \infty)$*, term* $t[i+2]$ *is computed as:*

$$t_{i+2} = t_i + (t_{i+1})^2$$

Given three integers, $t1$, $t2$, and n, compute and print the n^{th} term of a *modified Fibonacci sequence*.

Example

$t1 = 0$
$t2 = 1$
$n = 6$

- $t3 = 0 + 1^2 = 1$

- $t4 = 1 + 1^2 = 2$
- $t5 = 1 + 2^2 = 5$
- $t6 = 2 + 5^2 = 27$

Return 27.

Function Description

Complete the *fibonacciModified* function in the editor below. It must return the n^{th} number in the sequence.

fibonacciModified has the following parameter(s):

- *int t1*: an integer
- *int t2*: an integer
- *int n*: the iteration to report

Returns

- *int:* the n^{th} number in the sequence

Note: The value of $t[n]$ may far exceed the range of a 64-bit integer. Many submission languages have libraries that can handle such large results but, for those that don't (e.g., C++), you will need to compensate for the size of the result.

Input Format

A single line of three space-separated integers, the values of $t1$, $t2$, and n.

Constraints

- $0 \leq t1, t2 \leq 2$
- $3 \leq n \leq 20$
- t_n may far exceed the range of a 64-bit integer.

Sample Input

```
0 1 5
```

Sample Output

```
5
```

Explanation

The first two terms of the sequence are $t1 = 0$ and $t2 = 1$, which gives us a modified Fibonacci sequence of $\{0, 1, 1, 2, 5, 27, \ldots\}$. The 5^{th} term is 5.

CANDIDATE ANSWER

Language used: **Go**

```go
package main

import (
    "bufio"
    "fmt"
    "io"
    "os"
    "strconv"
    "strings"
    "math/big"
)



/*
```

```
 * Complete the 'fibonacciModified' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER t1
 *  2. INTEGER t2
 *  3. INTEGER n
 */

func fibonacciModified(t1, t2 int32, n int32) *big.Int {
    temp_arr := make([]*big.Int, n)
    temp_arr[0] = big.NewInt(int64(t1))
    temp_arr[1] = big.NewInt(int64(t2))

    for i := 2; i < int(n); i++ {
        temp_arr[i] = new(big.Int).Add(temp_arr[i-2],
new(big.Int).Mul(temp_arr[i-1], temp_arr[i-1]))
    }

    return temp_arr[n-1]
}

func main() {
    reader := bufio.NewReaderSize(os.Stdin, 16 * 1024 * 1024)

    stdout, err := os.Create(os.Getenv("OUTPUT_PATH"))
    checkError(err)

    defer stdout.Close()

    writer := bufio.NewWriterSize(stdout, 16 * 1024 * 1024)

    firstMultipleInput := strings.Split(strings.TrimSpace(readLine(reader)),
" ")

    t1Temp, err := strconv.ParseInt(firstMultipleInput[0], 10, 64)
    checkError(err)
    t1 := int32(t1Temp)

    t2Temp, err := strconv.ParseInt(firstMultipleInput[1], 10, 64)
    checkError(err)
    t2 := int32(t2Temp)

    nTemp, err := strconv.ParseInt(firstMultipleInput[2], 10, 64)
    checkError(err)
    n := int32(nTemp)

    result := fibonacciModified(t1, t2, n)

    fmt.Fprintf(writer, "%d\n", result)

    writer.Flush()
}

func readLine(reader *bufio.Reader) string {
    str, _, err := reader.ReadLine()
    if err == io.EOF {
        return ""
    }

    return strings.TrimRight(string(str), "\r\n")
}

```

```
79 func checkError(err error) {
80     if err != nil {
81         panic(err)
82     }
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0033 sec	3.31 KB
Testcase 2	Easy	Sample case	Success	0	0.0043 sec	3.37 KB
Testcase 3	Easy	Hidden case	Success	15	0.0064 sec	4.3 KB
Testcase 4	Easy	Hidden case	Success	15	0.011 sec	6.97 KB
Testcase 5	Easy	Hidden case	Success	15	0.0034 sec	5.49 KB
Testcase 6	Easy	Hidden case	Success	15	0.003 sec	3.4 KB
Testcase 7	Easy	Hidden case	Success	15	0.0135 sec	5.26 KB
Testcase 8	Easy	Hidden case	Success	15	0.0023 sec	5.22 KB
Testcase 9	Easy	Hidden case	Success	15	0.0028 sec	5.21 KB
Testcase 10	Easy	Hidden case	Success	15	0.0048 sec	3.35 KB

No Comments