Suppose I have a red block (R) of height 1, a blue block (B) of height 2, and a green block (G) of height 3. I can stack the blocks one above the other to build a tower. For example, I can stack in the order RBG and I would get a tower of height 6. I wish to compute how many different towers of a given height n can be built if there are ample blocks of each colour.

Let \( T(n) \) denote the number of towers of height \( n \).

Write a program that prints \( T(n) \) given \( n \).

**Sample Test Cases**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>274</td>
</tr>
<tr>
<td>21</td>
<td>223317</td>
</tr>
<tr>
<td>25</td>
<td>2555757</td>
</tr>
<tr>
<td>33</td>
<td>334745777</td>
</tr>
<tr>
<td>-20</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Numbers Part
2: Recursive Program (unit? unit=80&lesson=82)

Lecture 12: Virahanka Numbers Part
3: Iterative Program and Conclusion (unit? unit=80&lesson=83)

Lecture 13: Program Organization and Functions
Part 1: Introduction (unit? unit=80&lesson=84)

Lecture 13: Program Organization and Functions
Part 2: Splitting into files (unit? unit=80&lesson=85)

Lecture 13: Program Organization and Functions
Part 3: Namespaces (unit? unit=80&lesson=86)

Lecture 13: Program Organization and Functions
Part 4: How to use C++ without simplecpp (unit? unit=80&lesson=87)

Lecture 14: Advanced Features of Functions Part 1: Introduction and passing one function to another (unit? unit=80&lesson=88)

Test Case 8
2

Test Case 9
3

Test Case 10
4

Test Case 11
5

Test Case 12
-10

Test Case 13
29

The due date for submitting this assignment has passed.
As per our records you have not submitted this assignment.

Sample solutions (Provided by instructor)

```cpp
#include <iostream>
define repeat(x) for(int _iterator_i = 0, _iterator_limit = x; _iterator_i < _iterator_limit; _iterator_i++)
#define main_program int main()
define main_program
#include <cmath>
using namespace std;
main_program
{
int n; cin >> n;
int ti=1, tiplus1=2, tiplus2=4, res;

if(n == 1) res = 1;
else if(n == 2) res = 2;
else if(n == 3) res = 4;
else{
  int tiplus3=0;
  for(int iplus3=4; iplus3<=n; iplus3++){
    tiplus3 = ti+tiplus1+tiplus2;
    ti=tiplus1;
    tiplus1=tiplus2;
    tiplus2=tiplus3;
  }
  res = tiplus3;
}
cout << res << endl;
}
```

2/4
Lecture 14: Advanced Features of Functions Part 2: Lambda expressions (unit?unit=80&lesson=89)

Lecture 14: Advanced Features of Functions Part 3: Default values to parameters (unit?unit=80&lesson=90)

Lecture 14: Advanced Features of Functions Part 4: Function overloading and lecture conclusion (unit?unit=80&lesson=91)

Download Videos (unit?unit=80&lesson=182)

Weekly Feedback (unit?unit=80&lesson=194)

Quiz: Week 6 Assignment (assessment?name=215)

Week 6 Programming Assignment 1 (/noc20_cs53/progassignment?name=217)

Week 6 Programming Assignment 2 (/noc20_cs53/progassignment?name=218)

Week 7

Week 8

Week 9