Recursion

# 20465 מעבדה בתכנות מערכות‏[1](http://www.openu.ac.il/courses/20465.htm#remarks)

To download this document and all the oop courses clone it from git repository at:

If you don’t know git , just look at the google for git and cloning repository

git clone https://github.com/EliArad/OOP

Factorial!

How to look on recursion it is not much clear.

Takes the factorial

The definition for factorial is factorial (n) = n \* (n -1)

For example factorial of 5! = 5\*4\*3\*2\*1 = 5 \* 4!

Where

4! = 4 \*3!

3! = 3 \* 2!

2! = 2\* 1!

And 1! = 1.

How it is written in computer program in recursion:

int Factorial (int n)

{

if (n == 1)

return 1;

return n\* factorial(n - 1);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", factorial(5));

return 0;

}

First we call to Factorial(5) which is n \* Factorial (n-1) so let’s start the computer calls:

Factorial(5) = 5 \* Factorial (5-1)

Note that the computer will call to factorial of 4 (and it still does not know the answer of it)

Factorial(4) = 4 \* factorial(4-1)

Factorial(3) = 3 \* factorial(3-1)

Factorial(2) = 2 \* factorial(2-1)

Let’s write it again , but now with the numbers subtracted already:

Factorial(5) = 5 \* factorial(4)

Factorial(4) = 4 \* factorial(3)

Factorial(3) = 3 \* factorial(2)

Factorial(2) = 2 \* factorial(1) = 2

And factorial of 1 will return 1.

The important thing to understand here is that we want to know only the factorial of 5!

This is what we called on the main program.

Now that factorial(1) = 1 the program will know all its missing parts.

Factorial(5) = 5 \* factorial(4) we can write 24 here and get that factorial of 5 = 120

Factorial(4) = 4 \* factorial(3) we can write 6 here and get that factorial of 4 = 24

Factorial(3) = 3 \* factorial(2) we can write 2 here and get that factorial of 3 = 6

Factorial(2) = 2 \* factorial(1) we can write 1 here and get that factorial of 2 = 2

If we write the functions call each time we will get a list of calls that depends on each other.

Those calls are waiting to be solved by the stop condition.

Once the last call is solved, all other are solved and we can get our result.

Fibonacci

Fibonacci is a series that return the sum of a the two numbers before of that number starting from 0

For examples:

0 1 2 3 5 8 13 21 33 54 88

int Fibonacci(int n)

{

if (n == 0)

return 0;

if (n == 1)

return 1;

return Fibonacci (n - 1) + Fibonacci (n - 2);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

Fibonacci (5);

return 0;

}

Lets write as the function calls:

Fibonacci(5) = Fibonacci(4) + Fibonacci(3);

Fibonacci (4) = Fibonacci(3) + Fibonacci(2)

Fibonacci(3) = Fibonacci(2) + Fibonacci(1); // here Fibonacci(1) already return 1

Fibonacci(2) = Fibonacci (1) + Fibonacci (0) -- those two are the final state and we know that they are 1 and 0.

We want to know what is: Fibonacci (5) = Fibonacci(4) + Fibonacci(3)

Fibonacci(5) = Fibonacci(4) + Fibonacci(3); 3 + 2 = 5 and this is out answer

Fibonacci (4) = Fibonacci(3) + Fibonacci(2) = 2 + 1 = 3

Fibonacci(3) = Fibonacci(2) + Fibonacci(1); = 1 + 1 = 2

Fibonacci(2) = 1 + 0 = 1

Conclusion:

To solve recursion of that math problem we need to provide the function, formula in respect to its own.

To see all stage we just need to write the sequence of calls, and wait for our solver,

The solver is the final conditions where the function return a solution and stop call it self.

When we have all equation, function calls, we start solve from bottom up until we reached

Our requested main call.

Find out what this program prints:

Let’s take an example of a program that calculate something and lets try to understand what the function return before we execute it:

int findout(int n, int y)

{

if (n == y)

return y \* y;

return n \* findout(n - 1, y);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", findout(10, 3));

return 0;

}

findout(10, 3) = 10 \*findout(9, 3);

findout(9, 3) = 9 \* findout(8, 3);

findout(8, 3) = 8 \* findout(7, 3);

findout(7, 3) = 7 \* findout(6, 3);

findout(6, 3) = 6 \* findout(5, 3);

findout(5, 3) = 5 \* findout(4, 3);

findout(4, 3) = 4 \* findout(3, 3);

findout(3, 3) = 3 \* findout(2, 3);

Let’s write the solver from bottom to up:

findout(10, 3) = 10 \*findout(9, 3); 10 \* 9 \* 8 \* 7 \* 6 \* 5 \* 4 \* 9 = 5443200

findout(9, 3) = 9 \* findout(8, 3); 9 \* 8 \* 7 \* 6 \* 5 \* 4 \* 9

findout(8, 3) = 8 \* findout(7, 3); 8 \* 7 \* 6 \* 5 \* 4 \* 9

findout(7, 3) = 7 \* findout(6, 3); 7 \* 6 \* 5 \* 4 \* 9

findout(6, 3) = 6 \* findout(5, 3); 6 \* 5 \* 4 \* 9

findout(5, 3) = 5 \* findout(4, 3); 5 \* 4 \* 9

findout(4, 3) = 4 \* findout(3, 3); = 4 \* 9

findout(3,3) is the end , it does not call any more to findout

Another program:

int findout(int n, int y)

{

if (n == y)

return y \* y;

if (y < 0)

return -1;

if (n < 0)

return 0;

return (n - 1) + findout(n - 1, y) + findout(y - 1, n);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", findout(10, 3));

return 0;

}

Let’s starting by writing the steps, function calls and solve when we reached to the end

findout(10, 2) = (9) + findout(9, 2) + findout(1, 10);

findout(9, 2) = 8 + findout(8,2) + findout(1,9);

findout(8, 2) = 7 + findout(7,2) + findout(1,8);

findout(7, 2) = 6 + findout(6,2) + findout(1,6);

findout(6, 2) = 5 + findout(5,2) + findout(1,5);

findout(5, 2) = 4 + findout(4,2) + findout(1,4);

findout(4, 2) = 3 + findout(3,2) + findout(1,3);

findout(3, 2) = 2 + findout(2,2) + findout(1,2);

here we have findout(2,2) can be solve for y\*y = 2\*2 = 4;

findout(1,10) = 0 + findout(0 , 10) + findout(9,1);

findout(9,1) = 8 + findout(8,1) + findout(0,9);

findout(8,1) = 7 + findout(7,1) + findout(-1,8);

findout(7,1) = 6 + findout(6,1) + findout(-2,7);

findout(6,1) = 5 + findout(5,1) + findout(-3,6);

findout(5,1) = 4 + findout(4,1) + findout(-4,5);

findout(4,1) = 3 + findout(3,1) + findout(-5,4);

findout(3,1) = 2 + findout(2,1) + findout(-6,3);

findout(2,1) = 1 + findout(1,1) + findout(-7,2);

findout(0 , 10) = -1 + findout(-1 , 10) + findout(9 , 0);

findout(**9 , 0**) = 8 + findout(8,0) + findout(-1,9)

findout(**8 , 0**) = 7 + findout(7,0) + findout(-1,8)

findout(**7 , 0**) = 6 + findout(6,0) + findout(-1,7)

findout(**6 , 0**) = 5 + findout(5,0) + findout(-1,6)

findout(**5 , 0**) = 4 + findout(4,0) + findout(-1,5)

findout(**4 , 0**) = 3 + findout(3,0) + findout(-1,4)

findout(**3 , 0**) = 2 + findout(2,0) + findout(-1,3)

findout(**2 , 0**) = 1 + findout(1,0) + findout(-1,2)

findout(**1 , 0**) = 0 + findout(0,0) + findout(-1,1)

findout(**0 , 0**) = 0 + findout(-1,0) + findout(-1,0)

Now we can solve for all , we need of course start from the last calls, bottom up.

We reached to all stop conditions and we will be able to solve up to the real call.

findout(10, 2) = (9) + findout(9, 2) + findout(1, 10);

findout(9, 2) = 8 + findout(8,2) + findout(1,9);

findout(8, 2) = 7 + findout(7,2) + findout(1,8);

findout(7, 2) = 6 + findout(6,2) + findout(1,6);

findout(6, 2) = 5 + findout(5,2) + findout(1,5);

findout(5, 2) = 4 + findout(4,2) + findout(1,4);

findout(4, 2) = 3 + findout(3,2) + findout(1,3);

findout(3, 2) = 2 + findout(2,2) + findout(1,2);

here we have findout(2,2) can be solve for y\*y = 2\*2 = 4;

findout(1,10) = 0 + findout(0 , 10) + findout(9,1);

findout(9,1) = 8 + findout(8,1) + findout(0,9);

findout(8,1) = 7 + findout(7,1) + findout(0,8);

findout(7,1) = 6 + findout(6,1) + findout(0,7);

findout(6,1) = 5 + findout(5,1) + findout(0,6);

findout(5,1) = 4 + findout(4,1) + findout(0,5);

findout(4,1) = 3 + findout(3,1) + findout(0,4);

findout(3,1) = 2 + findout(2,1) + findout(0,3);

findout(2,1) = 1 + findout(1,1) + findout(0,2);

findout(1,1) = 0 + findout(0,1) + findout(0,1);

findout(0,1) = -1 + findout(-1,1) + findout(0,0);

findout(0 , 10) = -1 + findout(-1 , 10) + findout(9 , 0) = -1 + 0 + 8 + 17 – 1 = 23

findout(**9 , 0**) = 8 + findout(8,0) + findout(-1,9) = 8 + 17 - 1

findout(**8 , 0**) = 7 + findout(7,0) + findout(-1,8) = 7 + 6 + 7 – 1 – 1 = 18

findout(**7 , 0**) = 6 + findout(6,0) + findout(-1,7) = 6 + 7 - 1

findout(**6 , 0**) = 5 + findout(5,0) + findout(-1,6) = 5 + 4 + 3 + 2 -3 – 1 -1 – 1 – 1 = 7

findout(**5 , 0**) = 4 + findout(4,0) + findout(-1,5) = 4 + 3 + 2 -3 – 1 -1 - 1

findout(**4 , 0**) = 3 + findout(3,0) + findout(-1,4) = 3 + 2 -3 – 1 -1

findout(**3 , 0**) = 2 + findout(2,0) + findout(-1,3) = 2 -3 - 1

findout(**2 , 0**) = 1 + findout(1,0) + findout(-1,2) = 1 -3 – 1 = -3

findout(**1 , 0**) = 0 + findout(0,0) + findout(-1,1) = 0 +(-2) + (-1) = -3

findout(**0 , 0**) = -1 + findout(-1,0) + findout(-1,0) = -1 + 0 + (-1) = -2

אני לא יודע מה איתכם?

אבל אני התיאשתי באמצע כנראה מלחשב את הדבר הענק הזה

בוא נראה מהם הקריאות שהתוכנית הזו מריצה

8 + findout(8, 3) + findout(2 , 9)

7 + findout(7, 3) + findout(2 , 8)

6 + findout(6, 3) + findout(2 , 7)

5 + findout(5, 3) + findout(2 , 6)

4 + findout(4, 3) + findout(2 , 5)

3 + findout(3, 3) + findout(2 , 4)

1 + findout(1, 4) + findout(3 , 2)

0 + findout(0, 4) + findout(3 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 2) + findout(1 , 3)

0 + findout(0, 3) + findout(2 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 5) + findout(4 , 2)

0 + findout(0, 5) + findout(4 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

3 + findout(3, 2) + findout(1 , 4)

2 + findout(2, 2) + findout(1 , 3)

0 + findout(0, 3) + findout(2 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 4) + findout(3 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 6) + findout(5 , 2)

0 + findout(0, 6) + findout(5 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

4 + findout(4, 2) + findout(1 , 5)

3 + findout(3, 2) + findout(1 , 4)

2 + findout(2, 2) + findout(1 , 3)

0 + findout(0, 3) + findout(2 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 4) + findout(3 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 5) + findout(4 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 7) + findout(6 , 2)

0 + findout(0, 7) + findout(6 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

5 + findout(5, 2) + findout(1 , 6)

4 + findout(4, 2) + findout(1 , 5)

3 + findout(3, 2) + findout(1 , 4)

2 + findout(2, 2) + findout(1 , 3)

0 + findout(0, 3) + findout(2 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 4) + findout(3 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 5) + findout(4 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 6) + findout(5 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 8) + findout(7 , 2)

0 + findout(0, 8) + findout(7 , 1)

-1 + findout(-1, 8) + findout(7 , 0

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

6 + findout(6, 1) + findout(0 , 7)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

6 + findout(6, 2) + findout(1 , 7)

5 + findout(5, 2) + findout(1 , 6)

4 + findout(4, 2) + findout(1 , 5)

3 + findout(3, 2) + findout(1 , 4)

2 + findout(2, 2) + findout(1 , 3)

0 + findout(0, 3) + findout(2 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 4) + findout(3 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 5) + findout(4 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 6) + findout(5 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 7) + findout(6 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 9) + findout(8 , 2)

0 + findout(0, 9) + findout(8 , 1)

-1 + findout(-1, 9) + findout(8 , 0

7 + findout(7, 0) + findout(-1 , 8)

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

7 + findout(7, 1) + findout(0 , 8)

6 + findout(6, 1) + findout(0 , 7)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 8) + findout(7 , 0

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

7 + findout(7, 2) + findout(1 , 8)

6 + findout(6, 2) + findout(1 , 7)

5 + findout(5, 2) + findout(1 , 6)

4 + findout(4, 2) + findout(1 , 5)

3 + findout(3, 2) + findout(1 , 4)

2 + findout(2, 2) + findout(1 , 3)

0 + findout(0, 3) + findout(2 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 4) + findout(3 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 5) + findout(4 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 6) + findout(5 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 7) + findout(6 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 8) + findout(7 , 1)

-1 + findout(-1, 8) + findout(7 , 0

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

6 + findout(6, 1) + findout(0 , 7)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 10) + findout(9 , 2)

0 + findout(0, 10) + findout(9 , 1)

-1 + findout(-1, 10) + findout(9 ,

8 + findout(8, 0) + findout(-1 , 9)

7 + findout(7, 0) + findout(-1 , 8)

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

8 + findout(8, 1) + findout(0 , 9)

7 + findout(7, 1) + findout(0 , 8)

6 + findout(6, 1) + findout(0 , 7)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 8) + findout(7 , 0

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 9) + findout(8 , 0

7 + findout(7, 0) + findout(-1 , 8)

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

8 + findout(8, 2) + findout(1 , 9)

7 + findout(7, 2) + findout(1 , 8)

6 + findout(6, 2) + findout(1 , 7)

5 + findout(5, 2) + findout(1 , 6)

4 + findout(4, 2) + findout(1 , 5)

3 + findout(3, 2) + findout(1 , 4)

2 + findout(2, 2) + findout(1 , 3)

0 + findout(0, 3) + findout(2 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 4) + findout(3 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 5) + findout(4 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 6) + findout(5 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 7) + findout(6 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 8) + findout(7 , 1)

-1 + findout(-1, 8) + findout(7 , 0

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

6 + findout(6, 1) + findout(0 , 7)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

0 + findout(0, 9) + findout(8 , 1)

-1 + findout(-1, 9) + findout(8 , 0

7 + findout(7, 0) + findout(-1 , 8)

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

7 + findout(7, 1) + findout(0 , 8)

6 + findout(6, 1) + findout(0 , 7)

5 + findout(5, 1) + findout(0 , 6)

4 + findout(4, 1) + findout(0 , 5)

3 + findout(3, 1) + findout(0 , 4)

2 + findout(2, 1) + findout(0 , 3)

1 + findout(1, 1) + findout(0 , 2)

-1 + findout(-1, 2) + findout(1 , 0

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 3) + findout(2 , 0

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 4) + findout(3 , 0

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 5) + findout(4 , 0

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 6) + findout(5 , 0

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 7) + findout(6 , 0

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

-1 + findout(-1, 8) + findout(7 , 0

6 + findout(6, 0) + findout(-1 , 7)

5 + findout(5, 0) + findout(-1 , 6)

4 + findout(4, 0) + findout(-1 , 5)

3 + findout(3, 0) + findout(-1 , 4)

2 + findout(2, 0) + findout(-1 , 3)

1 + findout(1, 0) + findout(-1 , 2)

0 + findout(0, 0) + findout(-1 , 1)

**1199 התוצאה**

And why , again?

The program need to find all solvers that are defined by the stop conditions

if (n == y)

return y \* y;

if (y < 0)

return -1;

if (n < 0)

return 0;

For example

Findout(-1, x) , findout(0,0) , findout(9, -1)

And there are a lot of them

Stack overflow

What if I will remove the stop condition

if (n < 0)

return 0;

if we will print out all stages we will see that one of the solvers cannot be solved and continue **forever**.

This cause eventually a stack overflow and the program will crash

We should understand till now how the recoursion is working and how it calculate every stage untill it return the solution.

We also saw two real examples when we are providing a solution to a problem by the function it self and saw how the program behave until it solve the requested call.

**Now lets do some more real examples:**

כתוב פונקציה רקורסיבית לחישוב חזקה של מספר

int powerOf(int n, int y)

{

if (y == 0)

return 1;

if (y == 1)

return n;

return n \* powerOf(n, y - 1);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", powerOf(2, 5));

return 0;

}

How:

2**^**5 = 2\*2^4 = n\*n^n-1 = n \* powerOf(n, y-1)

נרשום את השלבים

powerOf(2,5) = 2 \* powerOf(2 , 4)

powerOf(2,4) = 2 \* powerOf(2 , 3)

powerOf(2 , 3) = 2 \* powerOf(2 , 2)

powerOf(2 , 2) =2 \* powerOf(2 , 1) 🡪 2 in power 1 is define in the stop condition as N

נציב:

powerOf(2,5) = 2 \* 2\*2\*2\*2 = 32 the result!!

powerOf(2,4) = 2 \* 2\*2\*2

powerOf(2 , 3) = 2 \* 2\*2

powerOf(2 , 2) =2 \* 2 🡪 2 in power 1 is define in the stop condition as N

כתוב פונקציה רקורסיבית המקבלת מערך ומחזירה את האיבר הגדול ביותר בו.

זוהי דוגמא לרקורסיה איטרטיבית.

הביצוע הפשוט היה פשוט לעבור פעם אחת על הכל ( שהרשימה לא ממוינת) ולמצוא את האיבר

הגדול ביותר.

התוכנית דואגת להכניס את כל איברי המערך לתוך המחסנית ורק בסיום מתחילה הבדיקה הרצויה.

הקוד שמעניין אותנו הוא הוא maxnum = וה return

הפעולה הזו דואגת ש maxnum יהי כל הזמן האחרון

int findArrayMaxNumber(int a[], int n)

{

int maxnum;

if (n == 1)

return a[0];

maxnum = findArrayMaxNumber(a + 1, n - 1);

if (maxnum > a[0])

return maxnum;

return a[0];

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int arr [9] = { 1, 5, 6, 100, 2, -3, 4, 5, 99 };

printf("%d\n", findArrayMaxNumber(arr, 9));

return 0;

}

Some notes:

1. Maxnum is a variable on stack , means it will be pushed for every call
2. The minimum of an array of one number , is the number it self ( n=1 min = a[0])

Very important note here:

We always care about the return value and this is our steps.

The function that call it self here is only a way of traversing the array and give use the last number.

On the way it push the array pointer a[0].

And the result is 100

if (maxnum > a[0]) // max = 100 , a[0] = 1

return maxnum;

return a[0];

if (maxnum > a[0]) // max = 100 , a[0] = 5

return maxnum;

return a[0];

if (maxnum > a[0]) // max become 100 , a[0] = 6

return maxnum;

return a[0];

if (maxnum > a[0]) maxnum = 99 , a[0] = 100

return maxnum;

return a[0]; return 100

if (maxnum > a[0]) maxnum = 99 , a[0] = 2

return maxnum;

return a[0];

if (maxnum > a[0]) maxnum = 99 , a[0] = -3

return maxnum;

return a[0];

if (maxnum > a[0]) maxnum = 99 , a[0] = 4

return maxnum;

return a[0];

if (maxnum > a[0]) // when we reached here , maxnum == 99 , a[0] = 5

return maxnum;

return a[0];

We start from here, bottom up^

כתוב פונקציה רקורסיבית המקבלת מחרוזת ותו בודד ומוצאת את מספר המופעים של התו במחרוזת

int findHowMany(char \*str, char p)

{

int count;

if (\*str == NULL)

return 0;

count = findHowMany(str+1, p);

if (p == \*str)

count++;

return count;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int x = findHowMany("abaa", 'a');

printf("%d\n", x);

return 0;

}

התרגיל הזה כמו קודמו עושה שני דברים

1. דואג שהמחרוזת תיכנס למחסנית ושנוציא אותה נקבל תו תו
2. הקוד שמעניין אותנו הוא ה return זה, נחשוב על זה כמו פונקציה שמחזירה ערך אבל הערכים שלה לא ידועים וצריך לחשב את כולם

נרשום את כל השלבים ונתחיל מהסוף למטה:

if (p == \*str) // count =2 , \*str = ‘a’

count++; // count = 3;

return count;

if (p == \*str) // count =2 , \*str = ‘b’

count++;

return count; // count still = 2

if (p == \*str) // count = 1, \*str = ‘a’

count++; 🡪 count = 2

return count;

// This is the place we finished call and starting to pop up from the stack

// count is 0 , \*str = ‘a’ , p is always ‘a’

if (p == \*str)

count++; 🡪 count = 1

return count;

פתח אלגוריתם רקורסיבי שיקבל מספר שלם וחיובי n ויחזיר את סכום ספרותיו.

int sumdigit(int n)

{

int sum = 0;

int count = 0;

if (n <= 0)

return 0;

count = sumdigit(n / 10);

count += n - (n / 10) \* 10;

return count;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", sumdigit(111111111));

return 0;

}

מה שמעניין בסוג זה של תרגילים הוא לא רק התנאי עצירה והפרוק של N אלא הדרך להשתמש ב count += בתוך הרקורסיה.

אם נדפיס אחרי count = sumdigit(n / 10); את N = נקבל:

1

11

111

1111

11111

111111

1111111

11111111

111111111

השורה הזו: n - (n / 10) \* 10; מפרקת את N ליחידות

:

נתונה הסדרה נתונה:

1 , 2 , 3 , 6 , 4 , 13 , 7 , 24 , 11 , 42 , ... :

שלושת האיברים הראשונים בסדרה הם: 3, 2, 1

כל איבר שנמצא במקום זוגי בסדרה, החל מהמקום ה-4 ,הוא סכום שלושת האיברים הקודמים לו.

כל איבר הנמצא במקום אי-זוגי בסדרה, החל מהמקום ה-5 ,הוא ההפרש המוחלט שבין שני האיברים במקומות הזוגיים שלפניו

. פתח אלגוריתם רקורסיבי לחישוב האיבר ה-n-י בסדרה זו (הקלט לפעולה יהיה המקום ה-n-י והערך שיוחזר יהיה ערכו של האיבר במקום זה).

אם לא ציינתי עד כה, אז הנה זה בא:

רקורסיה מסוג זה היא הקלה ביותר בסופו של דבר להבנה.

שימו לב שזה אותו רעיון כמו סידרת פיבונאצי. לפי הדרך שראינו בהתחלה נוכל לרשום את כל השלבים

אולם אחרי שהתנסנו גם בסידרת פיבונאצי וגם בשני תרגילים הנוספים אם ההצבה מלמטה למעלה

את השאלה הזו נוכל כבר לפתור לי ההגדרה של השאלה

#define abs(x) x < 0 ? -x : x

int sum(int n)

{

if (n == 1)

return 1;

if (n == 2)

return 2;

if (n == 3)

return 3;

if ((n % 2) == 0)

return sum(n - 1) + sum(n - 2) + sum(n - 3);

else

return abs(sum(n - 1) - sum(n - 3));

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", sum(10));

return 0;

}

רונן נמצא בראש סולם בעל n שלבים.

ברצונו לרדת מהסולם.

בכל פעם הוא יכול לרדת שלב בודד או שני שלבים.

פתח אלגוריתם רקורסיבי שיקבל כפרמטר את מספר השלבים בסולם ויחזיר את מספר הדרכים השונות בהן ניתן להגיע לתחתיתו.

שתי דרכים תקראנה "שונות" אם קיים לפחות קטע אחד השונה ביניהן. דוגמא: אם מספר 1 מייצג ירידה של שלב אחד, והמספר 2 מייצג ירידה של שני שלבים

.שונות דרכים שתי הן 1, 2 ,1 ,1 ,1 והדרך 2 ,1 ,1 ,1 ,1 א

char path[1000];

void printStairPath(char path[], int i, int n)

{

if (n<0)

return;

if (n == 0)

{

path[i] = '\0';

printf("%s\n", path);

return;

}

path[i] = '1';

printStairPath(path, i + 1, n - 1);

path[i] = '2';

printStairPath(path, i + 1, n - 2);

}

// It prints all the ways to climb up N step stairs

void printStairPaths(int n)

{

printStairPath(path, 0, n);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int n = 4;

printStairPaths(n);

return 0;

}

הפתרון הוא פיבונאצי אבל בשביל להדפיס את הדרך צריך לפצל את השורות של n-1 ו n-2

מה שאנחנו מחפשים זה לא את האיבר ה N אלא את מספר האפשרויות.

הדרך שבא יגיע המחשב לתנאי העצירה הם מספר האפשרויות שאנחנו מחפשים

פתח אלגוריתם רקורסיבי שיקבל שני מספרים טבעיים a ו-b ויחזיר את מכפלתם על ידי פעולות חיבור

(פעמים a\*b = a+a+...+a :

זאת שאלה מהזן הפשוט:

פונקציה שמגדירה את עצמה וגם מאד קל לכתוב את השלבים לפני שכותבים את התוכנית על מחשב כאשר נמצאים במיבחן בליעדיו.

int mult(int a, int b)

{

if (b == 1)

return a;

return a + mult(a, b - 1);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", mult(4, 5));

return 0;

}

mult(4,5) = 4 + mult(4 , 4);

mult(4,4) = 4 + mult(4, 3)

mult(4,3) = 4 + mult(4,2);

mult(4,2) = 4 + mult(4,1) = 4 + 4

נציב:

mult(4,5) = 4 + 4 + 4 + 4 + 4 = 20

mult(4,4) = 4 + 4 + 4 + 4

mult(4,3) = 4 + 4 + 4

mult(4,2) = 4 + mult(4,1) = 4 + 4

שאלה נתונה סדרת הנסיגה הכתובה בלשון מתמטית: פתח אלגוריתם רקורסיבי שיקבל מקום n בסדרה ויחזיר את האיבר ה-n-י בסדרה זו.



זהו שוב תרגיל סידרה לפונקציה שמגדירה את עצמה, נוכל לבדוק האם התוצאה נכונה

int calc(int n)

{

if (n == 1)

return 3;

return calc(n - 1) + 2;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

printf("%d\n", calc(10));

return 0;

}

Calc(10) = calc(9) + 2;

Calc(9) = calc(8) + 2;

Calc(8) = calc(7) + 2;

Calc(7) = calc(6) + 2;

Calc(6) = calc(5) + 2;

Calc(5) = calc(4) + 2;

Calc(4) = calc(3) + 2;

Calc(3) = calc(2) + 2;

Calc(2) = calc(1) + 2;

Calc(1) = 3

נציב ונפתור עבור n =10

Calc(10) = 3 + 2+ 2+ 2+ 2 + 2+ 2 + 2 + 2 + 2 = 21

Calc(9) = 3 + 2+ 2+ 2+ 2 + 2+ 2 + 2 + 2;

Calc(8) = 3 + 2+ 2+ 2+ 2 + 2+ 2 + 2;

Calc(7) = 3 + 2+ 2+ 2+ 2 + 2+ 2;

Calc(6) = 3 + 2+ 2+ 2+ 2 + 2;

Calc(5) = 3 + 2+ 2+ 2+ 2;

Calc(4) = 3 + 2+ 2+ 2;

Calc(3) = 3 + 2+ 2;

Calc(2) = 3 + 2;

Calc(1) = 3

פתח אלגוריתם רקורסיבי שיקבל מספר שלם חיובי n בבסיס 10 ויחזיר את ערכו בבסיס 2

First we need to know what the algorithm to convert decimal to binary is

<https://www.youtube.com/watch?v=H4BstqvgBow>

<https://www.youtube.com/watch?v=OezK_zTyvAQ>

<https://www.youtube.com/watch?v=XdZqk8BXPwg>

מקווה שראיתם הכל , נחמד לדעת כל מיני שיטות

השיטה השלישית עם החלוקה ל 2 היא הכי פשוט למימוש רקורסיבי.

int decimalToBinary(int num)

{

if (num < 1)

return 1;

decimalToBinary(num/2);

printf("%d", num % 2);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int c = decimalToBinary(244);

return 0;

}

קודם כל מחלקים הכל ב 2 ואחר כך מדפיסים עם השארית בעזרת פונקציית המודולו

כתוב פונקציה רקורסיבית המקבלת אורך צלע של ריבוע וצייר את הריבוע. דוגמא : עבור צלע באורך 4 יש לצייר את הריבוע:

void draw\_line(int width)

{

if (width == 0)

return;

draw\_line(width - 1);

printf("\*");

}

void \_draw\_rectangle(int width, int width1)

{

if (width == 0)

return;

draw\_line(width1);

printf("\n");

\_draw\_rectangle(width - 1, width1);

}

void draw\_rectangle(int width)

{

\_draw\_rectangle(width, width);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

draw\_rectangle(15);

return 0;

}

החשיבות בתרגיל הזה היא טכניקה.

איך מיישמים לולאה בתוך לולאה בצורה רקורסיבית?

באמצעות שתי פונקציות שונות וכתיבה מתודולוגית הפרדנו בין השורות.

פונקציה אחת מציירת שורה ופונקצייה שנייה קוראת ל draw\_line גם בצורה רקורסיבית.

זה נותן לנו להכניס שורת רווח ופתרון רקורסיבי נקי.

במקרים דומים של nesting loop שצריך לבצע באמצעות רקורסיה זו דרך טובה לפתרון.

Write a recursive function to reverse a string

void reverseString(char \*str)

{

if (\*str == 0)

return;

reverseString(str + 1);

printf("%c", \*str);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

reverseString("Hello world");

return 0;

}

Let’s write what will be inserting into the stack for each call

d -- top of the stack

ld

rld

orld

world

o world

lo world

llo world

ello world

Hello world

I am printing here what will be print out when the string will go out from the stack.

It always print the first character

d -- top of the stack

ld

rld

**o**rld

world

o world

lo world

llo world

ello world

Hello world – will be pushed first , but goes out last

יש עוד נושא ברקורסיה שנקרא **permutation**

שאלה שנשאלת בנושא זה היא string permutation

כלומר כתוב תוכנית שמדפיסה את כל האפשרויות של מחרוזת למשל ABC

מספר האפשרויות הם 6 3! =

הפתרון המלא שלא השאלה נמצא באתר הזה:

<http://www.geeksforgeeks.org/write-a-c-program-to-print-all-permutations-of-a-given-string/>

הוא ברור ומלא הסברים כולל סרטון הסברה.

A good video, almost lets you understands what we see here

<https://www.youtube.com/watch?v=PORo1ut9kMs>