FOR SSC AGENTS ONLY - DO NOT DISTRIBUTE

Stephenson Security Consultants
new recruit orientation
v1.023

Welcome to Stephenson Security.

For today's task you will be given 4 missions to complete. Each one will test your skills at programming and emphasize the skills for which you were recruited.

Use your mind and knowledge (conditionals, logical operators, random numbers, loops) to solve the puzzles and win.

Good luck candidates...

MISSION 1: Password Generator (20 pts)

There's a bit of a tradition here at SSC that the first thing a new agent does is to choose their password. However, instead of just picking something, we ask agents to write code to generate it.

Your task is to write a function that will generate a given number of passwords, with each one having the following characteristics:

1) 16 characters long
2) a random combination of lowercase letters, uppercase letters, and numbers

Example input: 5

Example output:

```
fMGlbe23o997P095
8ZkP6VT5J7rjeCkw
3h53gI6QEYt04b0r
7qVdaU0Xzkv36Qb0
4HhyHhBVG186AIJ4
```

MISSON 2: Device Duplication (20 pts)

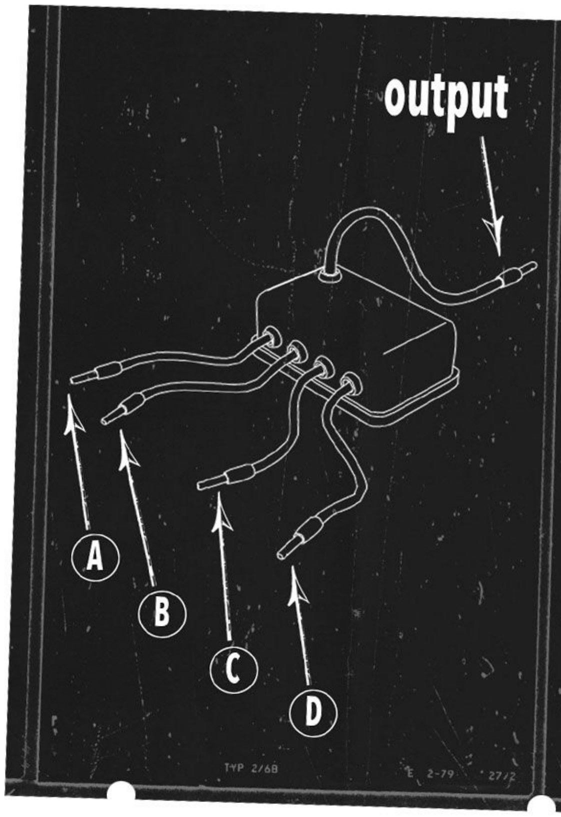
We've intercepted information regarding a device commonly used to secure certain facilities that we are interested in accessing.

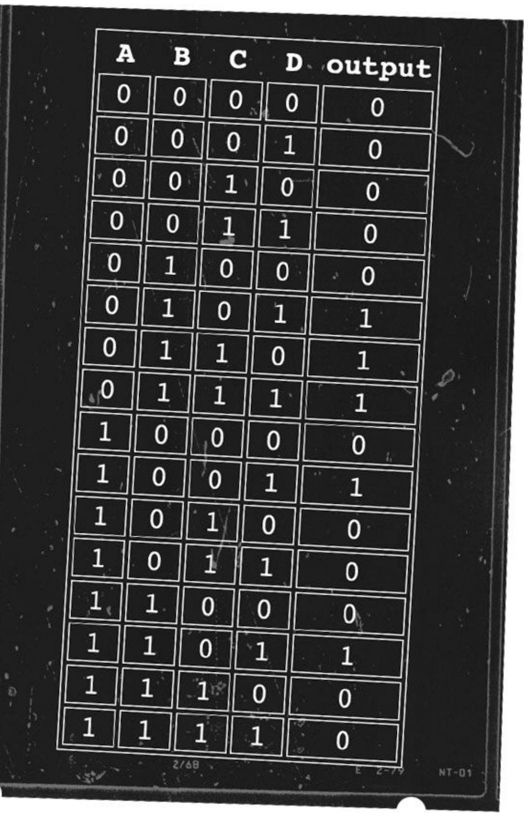
Unfortunately, we we're able to obtain one of the devices, but we do have information on its operation.

Here's what we know:

> The device takes four inputs, and produces a single output.
>
> Each of the inputs, as well as the output, is either on or off.

We need to create software that will mimic the operation of the device. We will be installing the software on a device of our own creation, for use in future infiltration missions.





A	B	C	D	output
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

2/68
E 2-79
NT-01

Your task is to write a function that will accept four inputs and produce a single output, according to the information in the above image.

Example input 1:

```
deviceDuplicate(true, true, true, false);
```

Example output 1:

```
0
```

Example input 2:

```
deviceDuplicate(true, false, false, true);
```

Example output 2:

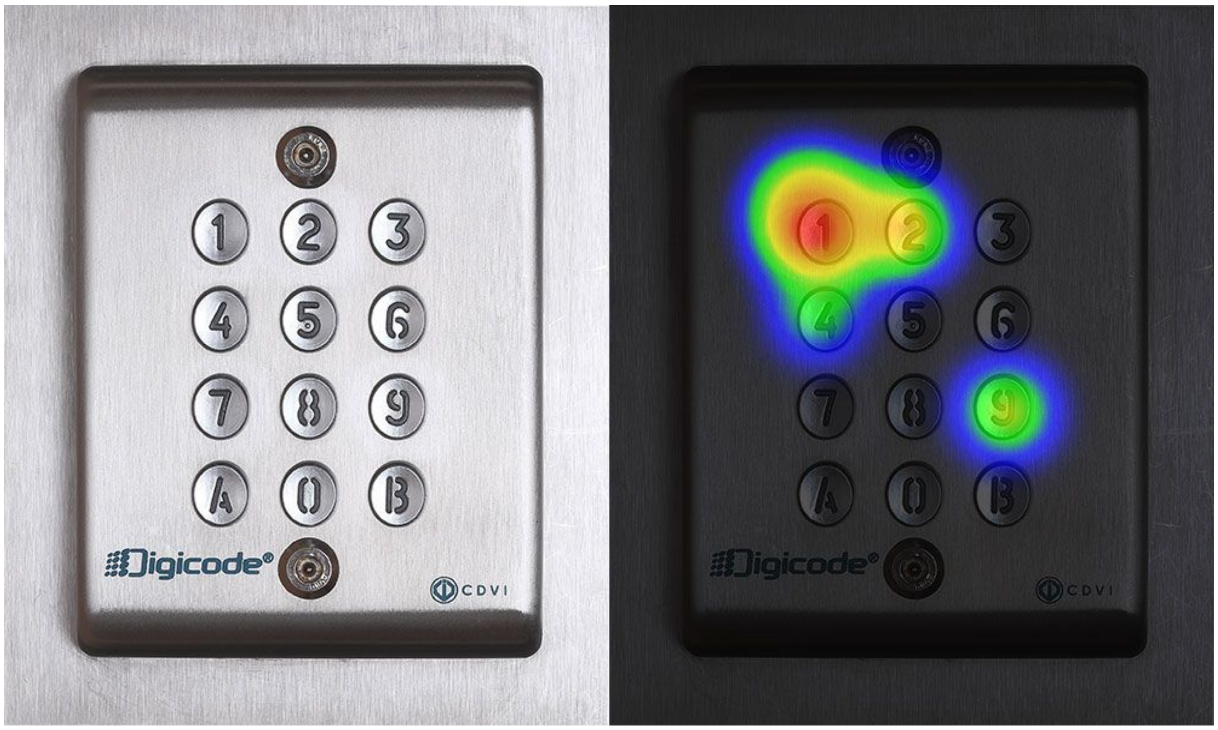
```
1
```

MISSION 3: Keypad Reader (30 pts)

Here at SSC, we are always looking for new ways to use technology to provide our agents with tactical advantages in the field. To that end, we have recently upgraded our standard-issue digital camera with infrared sensing capabilities.

We should be able to use that to determine a code entered into a recently-used keypad, since keys that are touched will retain heat.

Keys that have been touched more recently will be warmer, keys that were touched earlier will be slightly cooler, and keys that were not touched at all will be at room temperature.



Example raw image (left) and after being processed for leftover body heat (right), revealing that the numbers pressed were 4-9-2-1

We've already implemented software to map temperature values onto integers, but we need code that will take that data and determine the order in which the numbers were pressed.

Your task is to examine an array of integer values representing the buttons on a 3x3 keypad (in order). Each element in the array will be a value from 0 to 10, with higher values meaning more heat. A value of 0 means that the button was not pressed. Read through the array and find the values in increasing order to find which buttons were pressed and in which order.

The specific values will differ, but will always range between 0 and 10, with higher numbers meaning more heat (and more recent touches). Also, there will never be more than one occurrence of any particular number.

Example input:

```
     int keypadInfo[9] = {9,6,0,2,0,0,0,0,4};
```

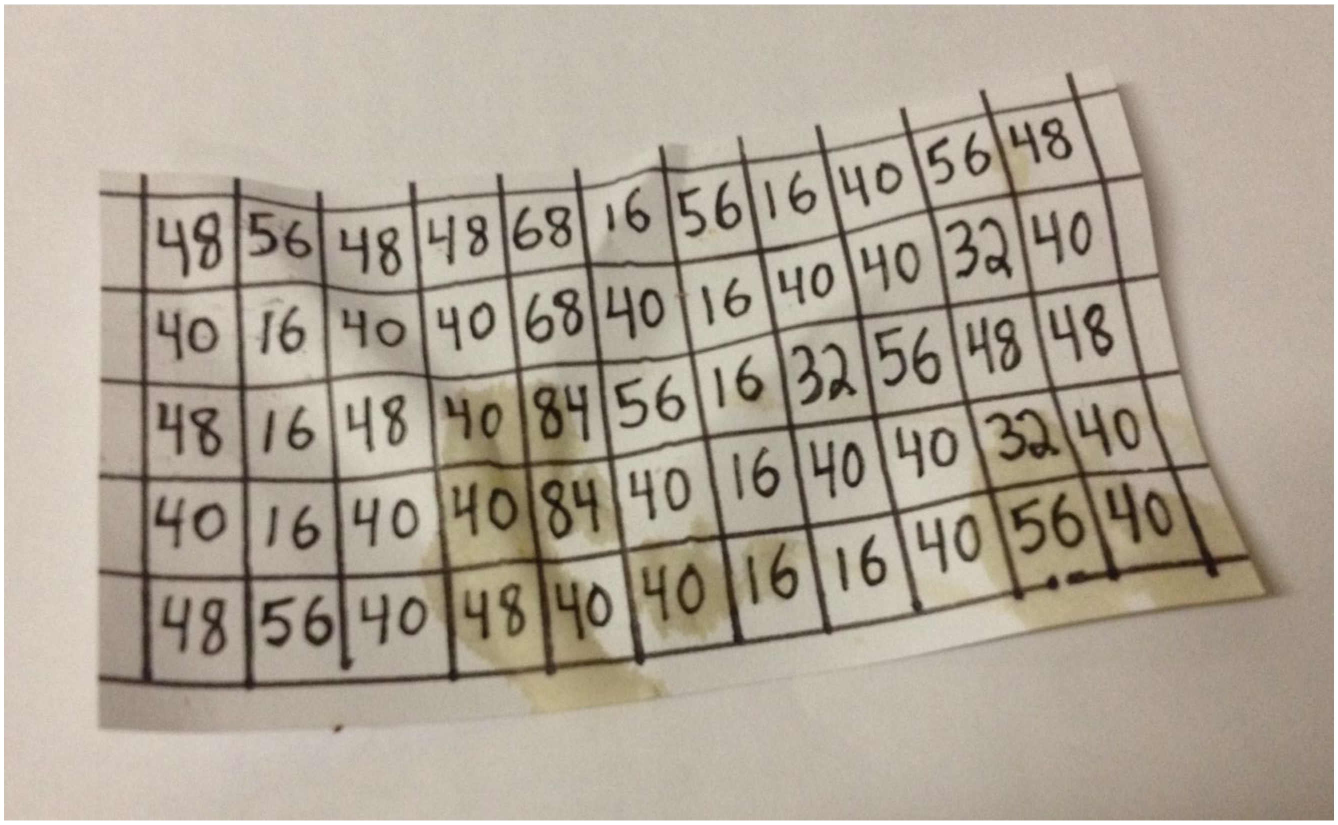
Example output:

```
     4921
```

In the above, the lowest non-zero value was index 3, indicating that the “4” button was the first one to be pushed. Continuing onward from there and finding increasing numbers gives us indices 8, 1, and 0, representing keys 9, 2, and 1.

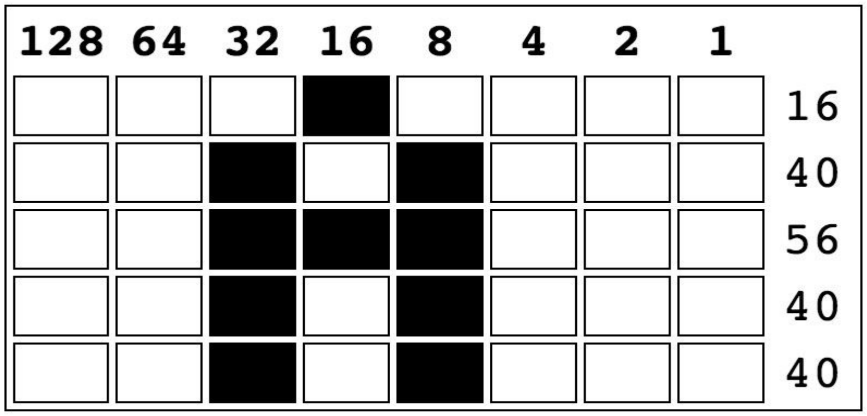
MISSION 4: the Intercepted Codename (30 pts)

We have come into the possession of a scrap of paper that we believe identifies the codename of an enemy agent. Your task is to take the numbers on the paper and determine the name of the agent.



48	56	48	48	68	16	56	16	40	56	48
40	16	40	40	68	40	16	40	40	32	40
48	16	48	40	84	56	16	32	56	48	48
40	16	40	40	84	40	16	40	40	32	40
48	56	40	48	40	40	16	16	40	56	40

We've seen this code before- it uses the binary representation of each number to visually encode letters, according to the following:



Write code that examines a two-dimensional array of integers. Each column represents a single letter. For each integer, print it out, using an underscore for each bit that is 0, and an asterix for each one that is 1.

So, if the number was 16, you would have 00010000, and would print:

```
___*____
```

If you do this for the entire input, you will get a word.

Example input:

```
    int threeLetters[5][3] = {{56, 24, 24},
        {32, 36, 36},
        {48, 36, 36},
        {32, 36, 36},
        {32, 24, 24}};
```

Example output:

```
__***______**______**___
__*_______*__*____*__*__
__**______*__*____*__*__
__*_______*__*____*__*__
__*________**______**___
```

Bonus Mission 1: location tracker (10 pts)

Sometimes, we need to identify locations based on partial information. We might know that a give place of interest is:

- North of a cafe
- East of an apartment building
- West of a subway station
- South of a restaurant

It is possible to represent a city as a two-dimensional grid of locations, with each location being assigned an integer representing the type of location it is.

Your task is to write code that can accept the IDs of the types of locations directly north, south, east, and west of an unknown location, then examine a grid to find the location that fits that criteria.

For example, given the following grid:

0	0	0	2	1
0	1	2	0	0
3	0	5	0	7
0	2	5	1	0
0	0	2	0	1

And the following information:
- there is a type 1 location due north of the target
- there is a type 2 location due south of the target
- there is a type 3 location due west of the target
- there is a type 5 location due east of the target

...your code would need to identify location 1, 2, since that is the only cell that meets all four criteria.

Example input:

```
    int cityGrid[5][5] = {
        {0, 0, 0, 2, 1},
        {0, 1, 2, 0, 0},
        {3, 0, 5, 0, 7},
        {0, 2, 5, 1, 0},
        {0, 0, 2, 0, 1}};

    findLocation(1, 2, 5, 3);
```

Example output:

```
    FOUND LOCATION: 1,2
```

Bonus Mission 2: the message (10 pts)

What is the message on the envelope that contained these documents?