Задачи на логику - дос

1. We want to make a row of bricks that is goal inches long. We have a number of small bricks (1 inch each) and big bricks (5 inches each). Return true if it is possible to make the goal by choosing from the given bricks. This is a little harder than it looks and can be done without any loops. See also: Introduction to MakeBricks

```
makeBricks(3, 1, 8) → true
makeBricks(3, 1, 9) → false
makeBricks(3, 2, 10) → true
```

2. Given 3 int values, a b c, return their sum. However, if any of the values is a teen -- in the range 13..19 inclusive -- then that value counts as 0, except 15 and 16 do not count as a teens. Write a separate helper "public int fixTeen(int n) {"that takes in an int value and returns that value fixed for the teen rule. In this way, you avoid repeating the teen code 3 times (i.e. "decomposition"). Define the helper below and at the same indent level as the main noTeenSum().

```
noTeenSum(1, 2, 3) → 6
noTeenSum(2, 13, 1) → 3
noTeenSum(2, 1, 14) → 3
```

3. Given 2 int values greater than 0, return whichever value is nearest to 21 without going over. Return 0 if they both go over.

```
blackjack(19, 21) → 21
blackjack(21, 19) → 21
blackjack(19, 22) → 19
```

4. Given 3 int values, a b c, return their sum. However, if one of the values is the same as another of the values, it does not count towards the sum.

```
loneSum(1, 2, 3) → 6
loneSum(3, 2, 3) → 2
loneSum(3, 3, 3) → 0
```

5. For this problem, we'll round an int value up to the next multiple of 10 if its rightmost digit is 5 or more, so 15 rounds up to 20. Alternately, round down to the previous multiple of 10 if its rightmost digit is less than 5, so 12 rounds down to 10. Given 3 ints, a b c, return the sum of their rounded values. To avoid code repetition, write a separate helper "public int round10(int num) {" and call it 3 times. Write the helper entirely below and at the same indent level as roundSum().

```
roundSum(16, 17, 18) → 60
roundSum(12, 13, 14) → 30
roundSum(6, 4, 4) → 10
```

6. Given 3 int values, a b c, return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.

```
luckySum(1, 2, 3) → 6
luckySum(1, 2, 13) → 3
luckySum(1, 13, 3) → 1
```

7. Given three ints, a b c, return true if one of b or c is "close" (differing from a by at most 1), while the other is "far", differing from both other values by 2 or more. Note: Math.abs(num) computes the absolute value of a number.

```
closeFar(1, 2, 10) → true
closeFar(1, 2, 3) → false
closeFar(4, 1, 3) → true
```

8. We want make a package of goal kilos of chocolate. We have small bars (1 kilo each) and big bars (5 kilos each). Return the number of small bars to use, assuming we always use big bars before small bars. Return -1 if it can't be done.

```
makeChocolate(4, 1, 9) → 4
makeChocolate(4, 1, 10) → -1
makeChocolate(4, 1, 7) → 2
```

9. Given three ints, a b c, one of them is small, one is medium and one is large. Return true if the three values are evenly spaced, so the difference between small and medium is the same as the difference between medium and large.

```
evenlySpaced(2, 4, 6) → true
evenlySpaced(4, 6, 2) → true
evenlySpaced(4, 6, 3) → false
```