TRAINING TASK LIBRARY TEAM 413

Powered by
Golang & Rust

Created by
Nikola Djurdjevic

For whom is this task library?

This library is for all people who are starting their software development, who are getting used to new environments in order to have the best possible adaptation, but also for experienced developers, who wants a good training.
Tasks are designed for Go and Rust environments, but they can be used for other environments as well.

Through algorithmic and modeling tasks, you will practice analytics and develop the ability to do a specific task as efficiently as possible, thinking about performance, resource consumption, the shortest path to a solution, but also about flexibility and abstraction, how a specific solution can be applied to other things.

Before you start to do anything, always draw multiple possible outcomes on diagram. Don't start immediately to write a code, especially for OOP tasks. Try to find the most efficient solution. Don't write first thing which you have on your mind, just to get the result. Remember, there is no easy or hard task. Task is a scenario which you can do in most efficient way, or you can't.

About Author

He is Golang/Rust Software Architect and he has a background in C++, Java and JavaScript. Development is built on Linux Systems and Darwin. Setup and administration for PostgreSQL, Mongo, Redis and RethinkDB.

Web applications on which he works are based on HTTP/1.1 or HTTP/2 Rest. Besides that, he also builds applications on native TCP, UDP and WebSocket traffic. Web architectures types are SAAS and FAAS.

For migration, integration, endpoint and Z testing he use the **T413** testing system which he created. T413 is the first testing tool that can test multiple microservices at the same time, regardless of traffic type and programming language in which architecture is implemented.

For running production and monitoring the system, he use the **SW413** system, which he created. SW413 is a native system background server that allows us to run, start, stop, remove or restart our service, monitor service usage and get service status. We won't have an additional delay or additional spent resources for our services as we would have with virtualization.

For system automation, he use the **AE413** system, which he created. The system is building automation through Web Agent. Automation can be executed with multiple types of configurations. There are multiple types of jobs that the system can execute. The setup and automation of the system are executed in a multithread environment, where performance and amount of spent resources have tremendously better productivity than any other automation system currently on market.

For scaling microservices, he uses the Load Balance system which he created. The system is called **LB413**. The system has 5 different modes. It can balance any backend traffic inside of any environment. Ten levels of security. Real-time update of all remote servers' statuses. Balance traffic without downtime.

Currently, in development, he has an **MS413** system, which is NoSQL in-memory database with the ability to write data on disk from active memory. The system is designed to replace Redis, with much better performance, with less spent resources, with the native scalable router, without any other 3rd party dependencies and with much more flexibility.

The developer is working on creating the environment for mass usage and enterprise-level for SW413, T413, AE413 and LB413 systems. Systems are optimized for Ubuntu, Mint, Manjaro, Fedora, RHEL8, Darwin(Intel) & Darwin(M1).
SW413 is also optimized for Windows. Whole 413 ecosystem is written in Go and Rust.

Developer is also writing training for developers. Training for Go you can watch for free on YouTube.

For all additional information you can contact developer on Discord: **Architect413#9400**.

Content

For algorithms there are 6 groups and each group posses 5 tasks.
For modeling tasks there are 6 tasks which you need to solve.

Algorithms – Group 1

1. Create function with one input, which will have string data type. In function input every two elements will be paired:

    CreateForm("David","9.20","Alex","8.10"...)

If we count from start David and 9.20 will be one pair.

In CreateForm create implementation where function input will be transformed into map[string]float64. Each name will be key in map, and each decimal number will be value of that key.

    Example: inputData := []string{"David","9.20","Alex","8.10","Max","6.20","Ben","7.50"}
      Output: map[string]float64{"David":9.20,"Alex":8.10,"Max":6.20,"Ben":7.50}

Note: Be careful about validation of data, converting of values and error checks.

---

2. We have two-dimensional list of float64. Create function which will have two input argument.

First argument will be [][]float64{{3.2,5.4},{6.3,1.4}} and second value will be float64.
The function must return all the elements that have a greater value then our input argument.
Also function must inform us in which list and on which place in that list we found that element.

As output of function we will return list of Message struct defined like this:

```
type Message struct {
        Value       float64
        List        int
        Position    int
}
```

where value is founded value, list is number of list, and position is position in the list.

Example:

    data := [][]float64{{3.2,5.4},{6.3,1.4},{2.5,6.5}}

      Case-1: FindAllGreaterThen(data,4.8) []Message
    Result-1: [{5.4,1,2} {6.3,2,1} {6.5,3,2}]

      Case-2: FindAllGreaterThen(data,5.5) []Message
    Result-2: [{6.3,2,1} {6.5,3,2}]

---

3. We have map[int][]float64.
Create function with 4 input arguments, where:

- **first argument** will be map[int][]float64 which we will pass as arguments.
  In this case that will be map which we already created.

- **second argument** will be []int. That list will help us find specific keys inside our created map

- **third argument** as string data type will be sign for specific operator which we want to use, like this:

    L - Less Than
    LE - Less Than or Equal to
    G - Greater than
    GE - Greater than or Equal to

  So third argument can be 4 values of data type string with values: L, LE, G, GE.

- **fourth argument** will be float64 data type, and with help of third argument we will have 4 scenarios:

Note: If 3. argument is L, and 4. argument is let's say 4.5, we need to find all keys from slice in first argument in our list of keys inside our map, where values of the keys from map which are connected to specific key are less than <4.5>.

Function must return output as map[int][]float64, where keys in map will be keys from slice of second argument in our function, which we were searching for, and values of these keys will be all values which satisfied our condition from 3. and 4. argument.

Note: If we searched for specific key, and we didn't found any value which satisfied our condition, we don't put that key in our output map. Example:

```
m1 := map[int][]float64{
        1:[]float64{3.8,4.6,5.2},
        2:[]float64{2.2,3.5,4.9},
        3:[]float64{2.7,3.1,4.1}
}

  Case-1: FindAll(m1,[]int{1,3},"G",4.0)
Result-1: map[int][]float64{1:{4.6,5.2},3:{4.1}}


  Case-2: FindAll(m1,[]int{2,3},"G",3.0)
Result-2: map[int][]float64{2:{3.5,4.9},3:{3.1,4.1}}


  Case-3: FindAll(m1,[]int{1,2,3},"LE",3.5)
Result-3: map[int][]float64{1:{},2:{2.2,3.5},3:{2.7,3.1}}
```

==========

4. We have [][]int with values:

```
[][]int{[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]}
```

We will take diagonal that begins on [0][0] and ends on [4][4]. Create two functions where:

- first function will sum all elements above diagonal
- second function will sum all elements bellow diagonal

==========

5. We have map[string]int with values: {"David":40,"Paul":20,"Bill":30,"Fred":50,"Alex":10}.
Create two functions where:

- first function will sort map based on keys of the map
- second function will sort map based on values of keys

Results must be form of map[string]int. Results:

```
{"Alex":10,"Bill":30,"David":40,"Fred":50,"Paul":20}
{"Alex":10,"Paul":20,"Bill":30,"David":40,"Fred":50}
```

==========

Algorithms – Group 2

==========

1. We have [][]int with values:

```
[][]int{[32,12,24,20],[18,40,22,30],[21,31,42,35]}
```

Create function which will give us back []int and values inside of that list, will be max value from each group inside of our [][]int. We have slice of 3 values for our 3 groups and each value is max value for a specific group.

```
Result: []int{32,40,42}
```

==========

2. We have []int like this:

list := []int{1,2,5,7,8,11,14}

Create function with one argument where argument will be int.
Function will put argument, on specific place, so slice will be still sorted.

Case-1: AddElement(list,4) Case-2: AddElement(list,9) Case-3: AddElement(list,12)
Result-1: [1,2,4,5,7,8,9,11,14] Result-2: [1,2,5,7,8,9,11,14] Result-3: [1,2,5,7,8,9,11,12,14]

==========

3. We have []int with values: 20,40,10,50,80,60,90,70,30,100.
We want to put them into [][]int in format of 3 groups, and sort them like this:

- first groups will be numbers which are <=30
- second groups will be numbers which are >30 and <=60
- third groups will be numbers which are >60 and <=100.

Result: [][]int{[10,20,30],[40,50,60],[70,80,90,100]}

==========

4. Create function with two input arguments. First argument will be [][]int with values:

[][]int[[6,11,17],[24,32,38],[45,50,55]]

Second argument will be []int with value: 15,30,11,49,50,32.
Each number from []int we want to put in specific range which that number satisfied.
Also we want to keep sorting, and we don't need duplicates inside of one group. Example:

list1 := [][]int{[[6,11,17],[24,32,38],[45,50,55]]} list2 := []int{15,30,11,49,50,32} AddElements(list1,list2)

Result: list1 := [][]int{[[6,11,15,17],[24,30,32,38],[45,49,50,55]]}

==========

5. Create function with 5 input arguments.
First argument will be []int with values: 6,45,17,81,32,55,95,50,24,72,62,38,28,68,33,89,11,49,77,99.

Second and third argument will be int. Second argument is lower boundary and third is higher boundary for searching elements in our first argument.

Fourth and fifth argument will define range from which we will separate our values from first argument.
For example, if our fourth argument is 40 and fifth argument is 70, groups will look like this:

- first group will be numbers with values <=40
- second group will be numbers with values >40 and <=70
- third group will be numbers with values >70

Possible values for second argument are 20 and 30. Possible values for third argument are 80 and 90.
Possible values for fourth argument are 40 and 60. Possible values for fifth argument are 50 and 70.

list := []int{6,45,17,81,32,55,95,50,24,72,62,38,28,68,33,89,11,49,77,99}

Example: CreateGroups(list,30,90,40,70)

LowerBoundary=30|HigherBoundary=90

Three groups separated in:
- first group will be numbers with values <=40
- second group will be numbers with values >40 and <=70
- third group will be numbers with values >70

Result: [][]int{[32,33,38][45,49,50,55,62,68][72,77,81,89]}

Algorithms – Group 3

1. Create function with 3 input variables. First two variables will be slice of ints, described like this:

p1 := []int{-4,5,-1,3,7,-3,6,-2,4,8,10,13,17,9,2} p2 := []int{12,7,13,8,3,14,6,15,5,4,1,-3,-5,-2,6}

Third input variable will be list of int. We will create pairs by index place, where first pair will be {-4,12}.
Function will return []int, where in that slice of ints we will add all index places of pairs from l1 and l2 slices
where sum of elements in a specific pair will be equal to any value from our third function input.

Case-1: FindSum(p1,p2,[]int{8,10}) Case-2: FindSum(p1,p2,[]int{9,11})
Result-1: []int{0,4,11,14} Result-2: []int{3,5,8,10}

2. Create function with 4 input variables. First two variables will be slice of ints, described like this:

p1 := []int{-4,5,-1,3,7,-3,6,-2,4,8,10,13,17,9,2} p2 := []int{12,2,7,6,3,8,0,15,5,-4,-1,-6,-5,-2,6}

Third input variable will be int. Fourth variable will be value for delta change declared as string.
We will create pairs by index place, where first pair will be {-4,12}.

Function will return []int, where in that list of ints we will add all index places of pairs from p1 and p2 lists
where sum of elements in specific pair will be equal to third input variable with delta change from fourth input variable.

Delta change value defines the deviation of the target value from the third input variable.
Delta value can be described in 3 different ways: positive value, negative value and absolute value.

For example, negative value must be less then "-1", positive value must be greater than "+1" and absolute value will be described with <*> mark.

For example, if target value from third input variable is 9 and delta value from fourth input variable is "*2", that means that we are targeting all values from pairs where elements have sum value >=7 && <=11, because of delta value.

Case-1: FindSum(p1,p2,9,+2) | Sum of values in range [>=9 && <=11] Result-1: []int{3,4,7,8,10}
Case-2: FindSum(p1,p2,8,-2) | Sum of values in range [>=6 && <=8] Result-2: []int{0,1,2,6,11,13,14}
Case-3: FindSum(p1,p2,11,*2) | Sum of values in range [>=9 && <=13] Result-3: []int{3,4,7,8,10,12}

3. We have []int in which we will put 60 random number in range >20 and <=40. This means that you need to generate 60 random numbers and put them in separated list, while they fit in condition >20 and <=40.

Create function which will separate elements from []int into 3 groups:

- first group will have elements which exists only on one place inside of our []int.
- second group will have elements which exists on two places inside of our []int.
- third group will have elements which exists on three on more places inside of our []int.

Example of generated slice: []int{23,25,27,40,38,27,31,27,19,25...}
Result of 3 groups: []int{23,40,28,31,29...} | []int{25...} | []int{27...}

In our slice list of 60 members, in this example, we have two times value 25 and three times value 27.

4. We have []float64 in which we will put 8 random number in range >1.0 and <=10.0. Float values inside of this slice will have precision 3. Variable b will have indexes for each element inside of our []float64 list presented as [][]float64.

```
list1 := []float64{3.213,2.543,6.435...}
list2 := [][]float64{
   {1.20,1.40}, {1.25,1.45}, {1.20,1.30}, {1.15,1.25}, {1.20,1.35}, {1.20,1.25}, {1.15,1.40}, {1.25,1.35},
}
list3 := []float64{5.5,4.5,6.5,8.5,14.5,12.5,9.5,11.5}
```

In this example indexes for value 3.213 are 1.20 and 1.40. We will have also variable c as []float64, where we will put limit values. Create function with 3 inputs, where arguments are variable a,b and c. Output variable of that function will be slice of empty interface. Inside of our function, variable c will be limit in following formula:

Value inside of list1 will be multiplied by all indexes connected to that value.
After we multiply values, will we get result.
If that result is smaller then limit, we will put inside of our output slice, that float64 value.
If that result is greater or equal then limit, we will put inside of our output, false value as boolean.

Example:

3.213*1.20*1.40 = 5.397 >> 5.397 is smaller then 5.5, so we will put in our output slice 5.397 value!
2.543*1.25*1.45 = 4.609 >> 4.609 is greater then 4.5, so we will put in our output slice false value!

Before execution of function, print values from first and second list.

==========

5. We have []int with values {39,50,64,81,72,31,43,48,29,99}.
Create function **CreatePairs()** where input variable will be previous created slide. Output of that function need to be int[][], where we will create pairs of of every first and last index place inside of slice and we will go like that until we end in the middle of the slice. Example of Result of **CreatePairs()** function:

[][]int{{39,99},{50,29},{64,48},{81,43},{72,31}}

Take that output as [][]int from **CreatePair()** function as new variable.

Create new function **CheckNormalization()** where first input variable will be output of **CreatePair()** function as [][]int. Second input variable in **CheckNormalization()** function will be int with specific assigned value described as normalization value.

From first input variable as [][]int, we will multiple members in each pair, and we need to check how much that value is different from normalization value which we assigned in **CheckNormalization()** as second input variable. For output of **CheckNormalization()** function we need to return []string where each string will explain analysis of each pair which we have as first input variable. In output we need to:

- describe which pair is analyzed, and what was the value of multiplication
- describe how much value which we get from multiplication is smaller or greater that normalization value

Example of task:

```
list := []int{39,50,64,81,72,31,43,48,29,99}
pairList := CreatePairs(list)
infoLog := CheckNormalization(pairList,3000)     > In this example, normalization value is 3000.
```

Example of output:

```
Analyzed pair have values: [39,99]. Multiplication value is greater then normalization value by 861.
Analyzed pair have values: [50,29]. Multiplication value is smaller then normalization value by 1550.
Analyzed pair have values: [64,48]. Multiplication value is greater then normalization value by 72.
Analyzed pair have values: [81,43]. Multiplication value is greater then normalization value by 483.
Analyzed pair have values: [72,31]. Multiplication value is smaller then normalization value by 768.
```

==========

Algorithms – Group 4

==========

1. We will create three different slices as []int where we will have following values:

list1 := {41,26,74,25,85,36,93,47,56,76,20,39}
list2 := {39,25,47,76,56,85,93,26,36,41,74,20}
list3 := {25,39,47,85,56,76,41,74,20,93,36,26}

Create function **CheckEquality()** where we need to analyze if all three lists have the same members. These three lists have the same members, in different order and **CheckEquality()** need to give us return value as boolean data type. Input of **CheckEquality()** will be three lists which we already defined.

==========

2. We will create []int where slice will have following values:

    list1 := []int{41,19,25,74,85,36,93,47,56,76,20,39,34,66,60,82,88,91,17,44,28,31,95,51,40,14}

Create function CheckAverageOfThreeElements() with two input variables.

First variable will be list1 slice and second input variable will be value 70 as data type int. Create all possible groups with 3 different elements from list1 slice and then report how many groups have average value greater then second input variable from CheckAverageOfThreeElements() which in this case is 70.

==========

3. We will create three different slices as []int where we will have following values:

    list1 := {41,26,74,25,85,36,93,47,56,76,20,39}
    list2 := {39,43,51,26,73,46,58,93,75,68,38,85}
    list3 := {93,26,70,29,85,36,80,79,50,42,20,39}

Create function TakeCommonValue() where we will have three input variables. Each input variable will be []int, and for input variables we will assign these three lists which we created in previous step.

Output of TakeCommonValue() function need to be []int where members of that slice will be all the same values which each of these three lists have. Output Result should be: []int{26,85,93,39}. These four values we have in all three lists.

==========

4. We will create three different slices as []int where we will have following values:

    list1 := {41,19,25,74,85,36,93,47,56,76,20,39}
    list2 := {39,43,56,66,32,46,58,93,74,22,81,29}
    list3 := {93,47,74,29,85,36,80,27,56,66,20,31}

Create function FindClosestPair() where input variables will be []int, our list1, list2 and list3 variables.
We will also have input variable with value 30 as data type int.

For output of FindClosestPair() function we need to find pair of values which are the closest to fourth input variable which in this case will be value 30. Output will be [][]int, where we will describe each closest pair for each list.

    Output Result should be: [][]int{{25,36},{32,29},{27,31}}

==========

5. We will create three different slices as []int where we will have following values:

    list1 := {41,19,25,74,85,36,93,47,56,76,20,39}
    list2 := {39,43,56,66,73,46,58,93,74,29,85,36}
    list3 := {93,47,74,29,85,36,80,25,56,66,20,39}

Create function TakeCommonPairs() where input variables will be []int, our list1, list2 and list3 variables.

Output of TakeCommonPairs() function need to be []string where members of that slice will
be description of same pair values which we founded in all three lists.
Pair of values here means two members of any list, which are side by side to each other.
For example, on start of first list [41,19] and [19,25] are first two pairs in list number 1.

We need to describe analysis of each pair we founded in multiple lists, their values and number of lists.
Don't write multiple times records if some pairs exist in all lists.
In this case, that pair will be [85,36], where we shouldn't write multiple analysis.
We will have only one analysis where info will be that pair exist in lists [1,2,3].

Output Result should be:

    Pair of values [85,36] exist in lists [1,2,3]. Pair of values [93,47] exist in lists [1,3].
    Pair of values [20,39] exist in lists [1,3]. Pair of values [56,66] exist in lists [2,3].
    Pair of values [74,29] exist in lists [2,3]. Pair of values [29,85] exist in lists [2,3].
    Pair of values [85,36] exist in lists [2,3].

Algorithms – Group 5

BE ADVISED: Built-in functions inside of language core are forbidden for usage in this group of tasks.

1. We will have two strings:

s1 = "abc bca cba acb abc bca cba acb"
s2 = "cba"

Create function with two inputs where s1 and s2 variables will be input values.
Function need to return []int where we need to store all index places where we find "cba" string value.

Function Example: FindMatch(s1, s2 string)
Expected Result: []int{8,24}

---

2. We will have three strings:

s1 = "abc bca cba acb abc bca cba acb"
s2 = "cba"
s3 = "acb"

Create function with three inputs where s1, s2 and s3 variables will be input values. Function need to return map[string]int where we need to store all index places for specific matched string value.

Inside of map[string]int, keys will be string value which we are searching and value for specific key will be all index places where we found that value.

Function Example:
FindMatch(s1, s2, s3 string)

Expected Result:
map[string]int{"cba": []int{8,24},"acb": []int{12,28}}

---

3. Create function with 2 inputs where s1 variable will be string and s2 will be []string.
Inside of s2 input variable we will store every string value which we want to find inside of s1 variable.

s1 = "acb abc bcacb cba acba abc bcacb cba acba abc bcacb qwe acb abc bcacb qwe acba"
s2 = []string{"cba","acba","bcacb","acb"}

We need to find all values from s2 variable inside of s1 variable and store them into structure Chain.
Structure chain will have following attributes:

ID = name of string which we are searching

List = multi-dimensional slice of int ([][]int) where we will store starting point index and ending point of index for specific value.

Function need to return slice of Chain structures as result. Function Example:

FindMatch(s1 string,s2 []string)
Expected Result:
[]Chain{
{ID:"cba",List: [][]int{[]int{11,14},[]int{30,33}}},
{ID:"acba",List: [][]int{.....}},
{ID:"bcacb",List: [][]int{.....}},
}

---

4. Create function with 2 inputs where s1 variable will be string and s2 will be []string. Inside of s2 input variable we will store pairs of string values. First value in each pair will define which value inside of s1 we need to replace, and second value will define that will be new value for replacement.

s1 = "abc bcacb cba acba abc bcacb cba acba abc bcacb qwe acb abc bcacb qwe acba"

Replacement-1: s2 := []string{"cba","Alex","acba","Ben","bcacb","David"}
Case-1: ReplaceValue(s1,s2)

Replacement-2: s2 := []string{"abc","Alex","qwe","Ben","cba","David"}
Case-2: ReplaceValue(s1,s2)

---

5. We have 3 string values, defined like this:

s1 := "zxc ert acba abcf dty ert acba abcd zxc qwe acb abcd nmv qwe acba"
s2 := "acba ghk bcacb ert acba abc sdf ert nmv bcacb dty qwe acb abcf ghk qwe"
s3 := "abcd bcacb ghk sdf nmv abc zxc sdf acba abcd sdf qwe acb abc dty qwe ghk bcacb"

Create function where we need to specify function argument as string values which we want to find inside our s1, s2 and s3 values. Function need to inform us, how many times we founded specific value, in which string value.

We need info which elements are found more than 3 times and which are not founded at all, if elements with those conditions exist. We need info about search ID and which values we are searching related to a specific search ID.

Search-Data-1: s4 = []string{"dty","vbn","nmv","ert"}
Case-1: FindValues(source,s4)

Search-Data-2: s4 = []string{"zxc","df","ghk","sdf","gbn","nmv","hk"}
Case-2: FindValues(source,s4)

Search-Data-3: s4 = []string{"abc","abcf", "abcd", "acb","acba","bcacb", "cf", "cd"}
Case-3: FindValues(source,s4)

Algorithms – Group 6

---

1. Create function which will search inside of multi-dimensional list for all unique values which we can find only on one place, and all elements which we will find in each dimension.

list1 := []int{7,9,40,85,18,8,99,31,14,105,48,22,10,38,12,60,41,21,115,15,6,33,20,17,13,35,75}
list2 := []int{70,100,31,60,90,41,55,33,115,99,50,75,40,65,38,59,35,45,58}
list3 := []int{15,45,60,18,99,55,72,50,38,7,65,119,70,95,6,115,48,58,75,10}
list4 := []int{85,90,105,95,38,115,60,100,99,125,75,119}

---

2. We have 3 lists of integers, separated into sectors.

list1 := []int{7,9,31,40,18,8,14,61,48,104,22,44,10,38,50,12,41,21,15,110,6,33,74,20,17,13,35,39,19,14,29}
list2 := []int{70,100,60,90,4,55,99,50,75,132,65,59,81,62,72,92,51,66,16,58,142,94,77,63,88,68,83,96}

list3 := []int{135,102,131,108,141,114,101,28,139,144,128,119,87,133,122,107,147,
105,150,124,109,121,34,136,103,123,95,143,106,127,117,125,112,120,130}

Each list belongs to a specific sector. In first sector we have numbers in range between 1 and 50.
In second sector are numbers between 51 and 100. In second sector are numbers between 101 and 150.

Create function which will find all missing values from each sector, where with them, we would have full sorted list inside of a specific sector. If there is any value, which doesn't belong to a specific sector, you must report that, and turn it back to sector where it belongs.

Note: Numbers which doesn't belong to right sector are: 4,16,28,34,61,74,87,95,104,110,132,142.

---

3. We have 3 different systems which are depending on each other, with labels S1,S1 and S3.
We have published production lines with versions like this:

==========
1.0.0|1.0.0|1.0.0
1.1.0|1.1.0|-----
1.1.1|-----|1.1.0
-----|1.2.0|1.1.1
1.2.0|1.3.0|-----
1.3.0|-----|1.2.0
1.3.1|1.4.0|1.2.1
-----|1.4.1|1.2.2
1.4.0|1.5.0|-----
-----|-----|1.3.0
1.5.0|1.6.0|1.3.1
1.5.1|1.6.1|-----
1.5.2|-----|1.4.0
-----|1.7.0|1.4.1
1.6.0|1.7.1|-----
1.6.1	1.7.2	1.5.0
1.7.0	1.8.0	1.6.0
1.7.1	1.8.1	1.6.1
1.7.2	-----	1.6.2
-----	1.9.0	1.6.3
1.8.0	1.9.1	-----
1.8.1	1.9.2	1.7.0
-----	1.9.3	1.7.1
1.9.0	-----	1.7.2
1.9.1	1.10.0	1.7.3
==========
First column describe versions for S1 system, second is for S2 and third is for S3. To have production line without any compatibility conflicts, we need to match compatible versions between 3 systems.

System S2 is depending on S1 and S3 is depending on S2. Each layer of versions is compatible with other system if versions are in the same update generation.

Second generation of S1 system have <1.2.0>, <1.3.0> and <1.3.1> versions. Compatible versions for second generation of S1 system for S2 system are <1.2.0> and <1.3.0> versions. Compatible versions for second generation of S2 system for S3 system are <1.1.0> and <1.1.1> versions.

Clients will ask us, how they can update or upgrade their infrastructure. They can send us current active versions or current active systems which they are using on their infrastructure. We need to inform them how they can update or upgrade systems depending that they have currently on their infrastructure.

If client side made a mistake in compatibility between systems, we need to inform client about that conflict. Also, dependencies between systems can't be broken. If client wants to install new version of S2 system and S1 system isn't already in use, that's a conflict.
==========
> Updates and upgrades descriptions
==========
Alex has active S1 v1.5.1 and he wants to install the newest version of S2 compatible with S1.
Ben has active S1 v1.7.2 and he wants to install the newest version of S2 and S3 compatible with S1.
Lana has active S1 v1.5.1 and she wants to update her current S2 v1.7.0 and S3 v1.4.0.

David has active S1 v1.8.1 and he wants to update his current S2 v1.9.3 and S3 v1.6.1.
Naomi has active S1 v1.8.0, S2 v1.9.1, S3 v1.6.1. She wants update for all systems.

Emma has active S1 v1.6.0. She wants to upgrade S1 to newest version of next S1 generation and to install the newest version of S2 and S3 compatible with that S1 version.

Helen doesn't have any of those systems in use. She wants to install the latest versions of all 3 systems.
Nolan is new user. He wants to install the S3 v1.6.3 and according to that S1 and S2 the latest compatible versions for that S3 version.
==========
NOTE: Create custom client side request and server side response.
==========
4. Create function as generator which will generate all possible combinations for all values from provided lists.
It's irrelevant how many lists are specified and how many values does each list have.

Generator need to be a single function as single abstract solution for any possible case related to this form.

Example-1:
List-1: 1|2|3
List-2: 3|5|7

Result-1:
{1,3},{1,5},{1,7},{2,3},{2,5},{2,7},{3,3},{3,5},{3,7}

Example-2:
List-1: 1|2
List-2: 3|5
List-3: 8|9

Result-2:
{1,3,8},{1,3,9},{1,5,8},{1,5,9},{2,3,8},{2,3,9},{2,5,8},{2,5,9}

==========

Example-3:
List-1: 1|2
List-2: 3|5|7
List-3: 4|6|8|9

Example-4:
List-1: 1|2|3|4
List-2: 5|6|7
List-3: 8|9

Example-5:
List-1: 1|2|3|4
List-2: 5|6
List-3: 7|8|9
List-4: 10|11

Example-6:
List-1: 1|2
List-2: 3|4|5|6
List-3: 7|8|9
List-4: 10|11|12|13
List-5: 14|15

Example-7:
List-1: 1|2|3|4|5|6
List-2: 7|8|9|10
List-3: 11|12
List-4: 13|14|15|16
List-5: 17|18|19|20|21
List-6: 22|23

==========

5. Create function as generator which will generate all possible combinations for all values from provided lists, under a specific list of conditions. Example:

List-1: 7|9|18|8|14|10|12|21|15|6|20|17|13
List-2: 31|41|33|40|38|35
List-3: 45|60|55|50|65|70|48|58|75
List-4: 85|90|105|95|115|100|99|125|119
List-5: 195|205|215|275|230|240|220|250|290|305

Inside of our generator function, we need to specify 4 inputs.
First input will be multi-dimensional slice of integers, where we will pass our lists, in multi-dimensional form.

Second input we be slice of integers where we want to specify which lists from multi-dimensional form we want to select for generation combinations. Lists will be selected with their index places, which means if we want to select first, second and fourth, we need to pass index places 0,1 and 3.

Third input will be target value for our combinations. Fourth input will be delta change (deviation) to our target value.

Delta change value defines the deviation of the target value from the third input variable.
Delta value can be described in 4 different ways: positive value, negative value, absolute value or zero.

Negative value must be less or equal then "-1", positive value must be greater or equal than "+1" and absolute value will be described with <*> mark. If delta is "0", than we won't target range for our combinations. We will target a specific number, which we can get from summing values from our combinations.

For example, if target value from third input variable is 50 and delta value from fourth input variable is "+10", that means that we are targeting all values from combinations where elements have sum value >=50 && <=60, because of delta value.

If target value from third input variable is 100 and delta value from fourth input variable is "*10", that means that we are targeting all values from combinations where elements have sum value >=90 && <=110, because of delta value.

If target value from third input variable is 120 and delta value from fourth input variable is "0", that means that we are targeting all values from combinations where elements have sum value equal to 120, because in this case we are not targeting range. We are targeting single specific value.

Generator need to be a single function as single abstract solution for any possible case related to this form. Example:

```
list1 := []int{7,9,18,8,14,10,12,21,15,6,20,17,13}
list2 := []int{31,41,33,40,38,35}
list3 := []int{45,60,55,50,65,70,48,58,75}
list4 := []int{85,90,105,95,115,100,99,125,119}
list5 := []int{195,205,215,275,230,240,220,250,290,305}
```

Examples of executed functions:

```
CreateCombinations(multiDimensionalForm,[]int{0,1},50,"-5")
CreateCombinations(multiDimensionalForm,[]int{0,1,2},100,"*10")
CreateCombinations(multiDimensionalForm,[]int{0,1,3},150,"+10")
CreateCombinations(multiDimensionalForm,[]int{0,1,3,4},350,"+30")
CreateCombinations(multiDimensionalForm,[]int{0,1,3,4},355,"0")
```

OOP Tasks

1. We have two technical positions: Developer and DevOps. Personal details for both positions will be:

- first name - last name - diploma - years of experience

We will also have information about Developer which languages and database he knows how to use.
We will also have information about DepOps which additional skills he has.

We need to create implementation where we can check if Developer or DevOps satisfy mandatory requirements for their technical position, so we can rank them on interview.

Developer needs to have at least 5 years of experience, he needs to know Golang as programming language and he needs to know how to use PostgreSQL and Mongo databases.

DevOps needs to have at least 5 years of experience and he needs to have 4 additional skills.
These skills are: Golang, Python, AWS, Mongo.

---

2. We want to create training list which we have inside of our gym.
When we create new training, we want to have following information:

- ID of training
- which coach is responsible for that training
- which set of exercises training contains
- time of training
- who is client for that private training
- length of training

When we create information about client and trainer, we want to know their:

- ID - first name - last name - phone number

For client we want to know two additional information, his height and weight. We need to create implementation where we can take data from each training. Following data will be:

- take coach data for specific training by ID
- take client data for specific training by ID
- take time when training will be started
- take exercise set for specific training
- update time of training, in case the timeline is changed
- update exercise set and length of training

In custom module, so can create custom lists of coaches and clients.
Create implementation for all cases which we need and execute them.

---

3. We want to create all information about music festival. That means that we need to describe list of concerts.
Each <concert> will have following information:

- ID - day of happening - stage ID - time of happening - artist ID

We will describe <artist> with following information:

- ID - name of the artist

We will describe <stage> with following information:

- ID - name of the stage - equipment ID

We will describe <equipment> with following information:

- ID - mix desk (equipment where they mix sound for concert)

Now we need to describe all people who are working on concert.
We will describe <staff> with following information:

- ID
- First Name
- Last Name
- StageID (on which stage they are working)
- Team (to which team they belong)
- Position (what is their technical position)

List of technical positions are:

- Video-E (Video Engineer)
- Audio-E (Audio Engineer)
- SpecialEffects-E (SpecialEffects Engineer)
- Video-T (Video Technician)
- Audio-T (Audio Technician)
- SpecialEffects-T (SpecialEffects Technician)

Every <stage> will have list of guards. We will describe <guard> with following information:

- ID
- First Name
- Last Name
- StageID (on which stage they are working)
- TeamID (to which team they belong)
- Sector (for which sector they are responsible)

==========

Create new lists in separated module where we will generate lists for artists, equipment and stages.
List of artists will be:

Artist{"1","Coldplay"} **Artist**{"2","Nightwish"} **Artist**{"3","Protoculture"}

List of equipment will be:

Equipment{"1","Soundcraft Vi3000"} **Equipment**{"2","Allen&Heath SQ-7"} **Equipment**{"3","Midas M32 Live"}

List of stages will be:

Stage{"1","Blue","1"} **Stage**{"2","Red","2"} **Stage**{"3","Green","3"}

List of concerts we will need to read from YAML file, which will look like this:

==========

```
- id: "1"
  day: 1
  stageID: "1"
  timeline: "2019-11-27T18:00:00Z"
  artistID: "1"
- id: "2"
  day: 1
  stageID: "2"
  timeline: "2019-11-27T18:00:00Z"
  artistID: "2"
- id: "3"
  day: 1
  stageID: "3"
  timeline: "2019-11-27T18:00:00Z"
  artistID: "3"
```

==========

Write this data in separated file and then in implementation read it.
List of guards we will need to read from JSON file, which will look like this:

```
==========================================================================================================
[
    {"id": 601,"firstName": "Paul","lastName": "Hoffman","stageID": "1","teamID": "1","sector": "1"},
    {"id": 602,"firstName": "Eliot","lastName": "Page","stageID": "1","teamID": "1","sector": "1"},
    {"id": 611,"firstName": "Darius","lastName": "Shaw","stageID": "1","teamID": "1","sector": "2"},
    {"id": 612,"firstName": "Freddie","lastName": "Hoffman","stageID": "1","teamID": "1","sector": "2"}
]
==========================================================================================================
```

Write this data in separated file and then in implementation read it.
List of staff we will need to read from CSV file, which will look like this:

```
==========================================================================================================
401;David;Hill;1;1;Video-E
402;Alex;Grant;1;1;Video-E
403;Jason;Milles;1;1;Video-T
404;Zak;Walker;1;1;Video-T
405;Luke;Jackson;1;1;Video-T
201;Samanta;Russo;1;1;Audio-E
202;James;Hawkins;1;1;Audio-E
203;Rhona;Davidson;1;1;Audio-T
204;Wade;Atkinson;1;1;Audio-T
205;Norma;Bender;1;1;Audio-T
801;Lillie;Wood;1;1;SpecialEffects-E
802;Paul;Russell;1;1;SpecialEffects-T
==========================================================================================================
```

Write this data in separated file and then in implementation read it.
In our implementation we want to have ability to take all data about specific staff who is
working on specific stage, and to take all information on which equipment is staff working on that concert.
We also want to have list of guards who is working on that concert, for that specific stage.

```
==========================================================================================
```

4. We want to create exchange office network. We will have 2 exchange offices with their exchange rates for a specific currencies. Offices can transform UK pounds, US dollars, EU euros, Russian ruble, Japanese yen and Indian rupee.

```
==========================================================================================
>>> Exchange Market 1
==========================================================================================
GBP -> USD (1-1.189)|GBP -> EUR (1-1.184)|GBP -> RUB (1-77.294)|GBP -> JPY (1-163.314)|GBP -> INR (1-94.480)
==========================================================================================
USD -> GBP (1-0.840)|USD -> EUR (1-0.995)|USD -> RUB (1-65.000)|USD -> JPY (1-137.265)|USD -> INR (1-79.435)
==========================================================================================
EUR -> GBP (1-0.844)|EUR -> USD (1-1.004)|EUR -> RUB (1-65.279)|EUR -> JPY (1-137.790)|EUR -> INR (1-79.768)
==========================================================================================
RUB -> GBP (1-0.012)|RUB -> USD (1-0.015)|RUB -> EUR (1-0.015)|RUB -> JPY (1-2.110)|RUB -> INR (1-1.222)
==========================================================================================
JPY -> GBP (1-0.006)|JPY -> USD (1-0.007)|JPY -> EUR (1-0.007)|JPY -> RUB (1-0.473)|JPY -> INR (1-0.578)
==========================================================================================
INR -> GBP (1-0.010)|INR -> USD (1-0.012)|INR -> EUR (1-0.012)|INR -> RUB (1-0.818)|INR -> JPY (1-1.727)
==========================================================================================
>>> Exchange Market 2
==========================================================================================
GBP -> USD (1-1.180)|GBP -> EUR (1-1.189)|GBP -> RUB (1-76.294)|GBP -> JPY (1-164.314)|GBP -> INR (1-93.480)
==========================================================================================
USD -> GBP (1-0.860)|USD -> EUR (1-0.999)|USD -> RUB (1-64.000)|USD -> JPY (1-136.265)|USD -> INR (1-80.435)
==========================================================================================
EUR -> GBP (1-0.824)|EUR -> USD (1-1.000)|EUR -> RUB (1-66.279)|EUR -> JPY (1-138.790)|EUR -> INR (1-80.768)
==========================================================================================
RUB -> GBP (1-0.082)|RUB -> USD (1-0.010)|RUB -> EUR (1-0.018)|RUB -> JPY (1-2.010)|RUB -> INR (1-1.122)
==========================================================================================
JPY -> GBP (1-0.004)|JPY -> USD (1-0.009)|JPY -> EUR (1-0.010)|JPY -> RUB (1-0.495)|JPY -> INR (1-0.555)
==========================================================================================
INR -> GBP (1-0.012)|INR -> USD (1-0.010)|INR -> EUR (1-0.011)|INR -> RUB (1-0.805)|INR -> JPY (1-1.740)
==========================================================================================
```

Exchange rates from markets needs to be transformed into JSON format file from previous table.
After that transformation, they need to be loaded from that JSON file into processing.

We have list of people who wants to exchange a specific amount of money from one currency to another.

Create processing where you will tell to person where to go for better exchange rates for as specific currency type, based on currency which they want to exchange.

Users accounts are described with Username, UserIDs and list of sectors transactions under a specific user account. Sector 0 is reserved for all money stored in USD, sector 1 is for GBP, sector 2 is for EUR, sector 3 is for RUB, sector 4 is for JPY and sector 5 is for INR. List of users accounts:

```
==========================================================================================
> Accounts|Username,AccountID,MoneyAmountForEachSector(USD,GBP,EUR,RUB,JPY,INR)
==========================================================================================
Alex|2453425652|50000|60000|70000|5000000|6000000|8000000
Ben|8674652543|60000|70000|80000|6000000|7000000|10000000
David|8562953754|30000|40000|30000|3000000|4000000|20000000
Emma|848563421|10000|10000|10000|1000000|1000000|10000000
Fiona|335647689|5000|5000|5000|100000|100000|1000000
Gina|242564578|4000|4000|4000|50000|50000|500000
Alex|667783546|0|0|0|0|0|0
Tina|565283951|0|0|0|0|0|0
Simon|4677859941|500000|600000|700000|0|0|0
Selena|4877639720|250000|350000|450000|0|0|0
Nolan|7445668990|0|0|0|2500000|4500000|7500000
Naomi|352678541|0|0|0|7500000|8500000|3500000
Paul|9122387545|10000000|10000000|10000000|0|0|0
Lana|8766234490|750000|2000000|10000000|0|0|0
Nolan|2556779340|0|2500000|2500000|0|0|0
==========================================================================================
```

List of transactions is:

```
==========================================================================================
> TransactionList|Username,AccountID,ListOfTransactions
==========================================================================================
Alex|2453425652|GBP-USD-500000
Ben|8674652543|GBP-EUR-750000
David|8562953754|USD-EUR-800000
Ema|848563421|USD-JPY-900000
Fiona|335647689|EUR-RUB-1000000
Gina|242564578|EUR-INR-1200000
Alex|667783546|RUB-GBP-15000000
Tina|565283951|RUB-USD-17500000
Simon|4677859941|JPY-GBP-25000000
Selena|4877639720|JPY-EUR-30000000
Nolan|7445668990|INR-RUB-5000000
Naomi|352678541|INR-JPY-8000000
Paul|9122387545|USD-EUR-650000|GBP-EUR-850000
Lana|8766234490|RUB-GBP-16500000|RUB-USD-19500000
Nolan|2556779340|JPY-GBP-32000000|JPY-EUR-36500000
==========================================================================================
```

New converted amount of money must be sent to user bank account in a specific account sector depending upon which currency, user is converting. Each exchange query need to have ID, request person name, exchange currencies for a specific transaction and the best exchange rate for a specific transformation.

Also implement logs and print output in command line in JSON format.

```
==========================================================================================
```

5. We want to create medicine store network. Inside of each store we will have list of medicine, with description tags:

> ID
> Medicine Name
> Purpose
> Level of Effectiveness (1-3)
> Cost in pounds
> Availability

For purpose we will have 6 types and types can be:

> Flu (Type-1)
> Throat-Infection (Type-2)
> Nose-Drops (Type-3)
> Stomach-Virus (Type-4)
> High-Pressure (Type-5)
> Headache (Type-6)

In store table types are described as T1,T2,T3,T4,T5 and T6. Level of Effectiveness can be number value from 1 to 3.

Currency for cost is pound. Availability is described as number of boxes.

```
=============================================================================================================
# ID | Medicine Name | Purpose | Level of Effectiveness (1-3) | Cost in pounds | Availability
=============================================================================================================
=========================
>>> Medicine Market 1
=========================
1 N13 T1 2 14 10
2 M10 T4 3 12 8
3 K21 T6 2 10 7
4 PL1 T3 3 18 7
5 TR1 T2 2 14 6
6 D10 T5 1 11 4
7 Q15 T6 3 20 3
8 L20 T3 2 15 8
9 GF1 T1 1 12 9
10 BC1 T5 2 14 5
11 ZC1 T4 1 14 6
12 MGS T2 2 12 12
13 W15 T5 3 20 4
14 H25 T2 2 15 7
15 AF1 T6 1 12 6
16 DFC T3 2 14 7
17 KP1 T4 3 14 9
18 TUP T1 2 12 8
=========================
>>> Medicine Market 2
=========================
1  FR2 T1 2 15 10
2  FT3 T4 3 13 6
3  QE1 T6 1 11 9
4  GH1 T3 3 19 7
5  TY2 T2 2 13 5
6  V10 T5 1 12 6
7  AD3 T6 3 21 5
8  LP2 T3 2 16 10
9  BK1 T1 1 13 6
10 ML2 T5 2 13 7
11 DV1 T4 3 12 8
12 MG2 T2 2 11 10
13 WR1 T5 3 19 3
14 HR2 T2 2 14 4
15 AU1 T6 1 10 6
16 DF2 T3 2 15 7
17 ZR3 T4 1 16 6
18 MFR T1 2 14 5
=================================================
```

For prescriptions we have specific query list:

```
=============================================================================================================
```

Alex need info about all medications related to Flu where level of effectiveness is 3.
Price is no more then 15 pounds.

Ben wants to buy any medication related to Throat-Infection where level of effectiveness is 2.
Price is between 10 and 13 pounds. He needs 2 boxes.

David wants to buy any medication related to Nose-Drops where level of effectiveness is 2
Price is no more then 16 pounds. He needs 2 boxes.

Elena wants to buy any medication related to Stomach-Virus where level of effectiveness is 2.
Price is no more then 16 pounds. She needs one box.

Fred wants to buy any medication related to High-Pressure where level of effectiveness is 3
Price is no more then 19 pounds. He needs 3 boxes.

Kyle wants to buy 3 boxes of AU1 and 3 boxes of AD3.

Merry wants to buy 2 boxes of K21 and 4 boxes of W15.

Kyle wants to buy 3 boxes of Q15 and 3 boxes of AD3. If you find one medicine, execute purchase anyway.

Selena wants to buy anything related to Stomach-Virus where level of effectiveness is 3 and it's not DV1 or KP1. Price is irrelevant. She needs 3 boxes.

==========

Be advised: People who need info about a specific medications, they didn't buy medicine at that moment, which means that Availability for a specific medicine on the specific market didn't got updated.

On every direct purchase, availability for the medication that was purchased, must be updated.
On update system need to inform us, if market on a specific purchase for a specific medication is out of stock.

For each prescriptions where purchase has been made, you need to find best solution for buyer in meaning of price.
On purchase you all need to print whole order with description, name of buyer and final order price.

If buyer want multiple medicines, and you don't find them all, you need to specify in query, if you want to finish order and execute purchase, even if you didn't found all items in store.

If we can't find a specific medicine, regardless if people just want info about medicine or they want to purchase it, system need to inform us about that.

==========

6. We will create hardware store market. Inside of each store we will have sectors of hardware components. Sectors will be:

> CPU	> Motherboard	> GPU	> RAM
> PowerSupply	> PcCase	> Cooler	> SSD

Each component has attributes according to component type. Markets have different prices and different stock.

We will load data to our markets from following database.
NOTE: Copy entire data to single file and then parse all data to structures. **Markets data is finished on page 24.**

==========

>MarketID=1

==========

CPU|ID,Maker,ChipSocket,Generation,ChipModel,ChipVersion,CoresNumber,BaseFrequency,TurboFrequency,Price,Stock

==========

1|INTEL|1200|11|i5|11600|6|2.8|4.6|250|0
2|INTEL|1200|11|i5|11600K|6|3.8|4.9|300|6
3|INTEL|1200|11|i7|11700|8|2.5|4.5|400|0
4|INTEL|1200|11|i7|11700K|8|3.6|4.8|450|7
5|INTEL|1200|11|i9|11900|8|3.9|4.9|480|5
6|INTEL|1200|11|i9|11900K|8|3.6|5.3|540|5
7|INTEL|1700|12|i5|12600|6|3.3|4.8|320|0
8|INTEL|1700|12|i5|12600K|10|3.8|4.9|420|9
9|INTEL|1700|12|i7|12700|12|2.1|4.9|460|6
10|INTEL|1700|12|i7|12700K|12|3.6|5.0|530|10
11|INTEL|1700|12|i9|12900|16|2.2|5.1|620|8
12|INTEL|1700|12|i9|12900K|16|2.6|5.2|700|5
13|AMD|AM4|3000|Ryzen7|3700X|8|3.6|4.4|260|7
14|AMD|AM4|3000|Ryzen9|3900X|12|3.8|4.6|500|6
15|AMD|AM4|5000|Ryzen5|5600|6|3.5|4.4|220|0
16|AMD|AM4|5000|Ryzen5|5600X|6|3.7|4.6|260|11
17|AMD|AM4|5000|Ryzen7|5700X|8|3.4|4.6|320|10
18|AMD|AM4|5000|Ryzen7|5800X|8|3.8|4.7|390|9
19|AMD|AM4|5000|Ryzen9|5900|12|3.7|4.8|500|8
20|AMD|AM4|5000|Ryzen9|5950X|16|3.4|4.9|700|6
21|AMD|AM5|7000|Ryzen5|7600X|6|4.7|5.3|380|14
22|AMD|AM5|7000|Ryzen7|7700X|8|4.5|5.4|520|12
23|AMD|AM5|7000|Ryzen9|7900X|12|4.7|5.6|700|10
24|AMD|AM5|7000|Ryzen9|7950X|16|4.7|5.7|900|8

==========

>MarketID=2

==========

CPU|ID,Maker,ChipSocket,Generation,ChipModel,ChipVersion,CoresNumber,BaseFrequency,TurboFrequency,Price,Stock

==========

1|INTEL|1200|11|i5|11600|6|2.8|4.6|260|5

2|INTEL|1200|11|i5|11600K|6|3.8|4.9|290|6
3|INTEL|1200|11|i7|11700|8|2.5|4.5|410|4
4|INTEL|1200|11|i7|11700K|8|3.6|4.8|440|7
5|INTEL|1200|11|i9|11900|8|3.9|4.9|490|5
6|INTEL|1200|11|i9|11900K|8|3.6|5.3|530|0
7|INTEL|1700|12|i5|12600|6|3.3|4.8|310|8
8|INTEL|1700|12|i5|12600K|10|3.8|4.9|430|9
9|INTEL|1700|12|i7|12700|12|2.1|4.9|450|6
10|INTEL|1700|12|i7|12700K|12|3.6|5.0|540|0
11|INTEL|1700|12|i9|12900|16|2.2|5.1|610|8
12|INTEL|1700|12|i9|12900K|16|2.6|5.2|710|5
13|AMD|AM4|3000|Ryzen7|3700X|8|3.6|4.4|270|7
14|AMD|AM4|3000|Ryzen9|3900X|12|3.8|4.6|490|6
15|AMD|AM4|5000|Ryzen5|5600|6|3.5|4.4|230|12
16|AMD|AM4|5000|Ryzen5|5600X|6|3.7|4.6|250|11
17|AMD|AM4|5000|Ryzen7|5700X|8|3.4|4.6|330|0
18|AMD|AM4|5000|Ryzen7|5800X|8|3.8|4.7|380|9
19|AMD|AM4|5000|Ryzen9|5900|12|3.7|4.8|520|8
20|AMD|AM4|5000|Ryzen9|5950X|16|3.4|4.9|720|0
21|AMD|AM5|7000|Ryzen5|7600X|6|4.7|5.3|400|14
22|AMD|AM5|7000|Ryzen7|7700X|8|4.5|5.4|500|12
23|AMD|AM5|7000|Ryzen9|7900X|12|4.7|5.6|680|0
24|AMD|AM5|7000|Ryzen9|7950X|16|4.7|5.7|880|0
==========
>MarketID=3
==========
CPU|ID,Maker,ChipSocket,Generation,ChipModel,ChipVersion,CoresNumber,BaseFrequency,TurboFrequency,Price,Stock
==========
1|INTEL|1200|11|i5|11600|6|2.8|4.6|240|5
2|INTEL|1200|11|i5|11600K|6|3.8|4.9|310|6
3|INTEL|1200|11|i7|11700|8|2.5|4.5|390|0
4|INTEL|1200|11|i7|11700K|8|3.6|4.8|460|0
5|INTEL|1200|11|i9|11900|8|3.9|4.9|470|5
6|INTEL|1200|11|i9|11900K|8|3.6|5.3|550|5
7|INTEL|1700|12|i5|12600|6|3.3|4.8|330|8
8|INTEL|1700|12|i5|12600K|10|3.8|4.9|410|9
9|INTEL|1700|12|i7|12700|12|2.1|4.9|470|6
10|INTEL|1700|12|i7|12700K|12|3.6|5.0|520|10
11|INTEL|1700|12|i9|12900|16|2.2|5.1|630|0
12|INTEL|1700|12|i9|12900K|16|2.6|5.2|690|0
13|AMD|AM4|3000|Ryzen7|3700X|8|3.6|4.4|260|7
14|AMD|AM4|3000|Ryzen9|3900X|12|3.8|4.6|510|0
15|AMD|AM4|5000|Ryzen5|5600|6|3.5|4.4|210|0
16|AMD|AM4|5000|Ryzen5|5600X|6|3.7|4.6|270|11
17|AMD|AM4|5000|Ryzen7|5700X|8|3.4|4.6|310|10
18|AMD|AM4|5000|Ryzen7|5800X|8|3.8|4.7|400|9
19|AMD|AM4|5000|Ryzen9|5900|12|3.7|4.8|480|8
20|AMD|AM4|5000|Ryzen9|5950X|16|3.4|4.9|690|6
21|AMD|AM5|7000|Ryzen5|7600X|6|4.7|5.3|410|14
22|AMD|AM5|7000|Ryzen7|7700X|8|4.5|5.4|530|12
23|AMD|AM5|7000|Ryzen9|7900X|12|4.7|5.6|690|10
24|AMD|AM5|7000|Ryzen9|7950X|16|4.7|5.7|910|8
==========
>MarketID=1
==========
Motherboard|ID,Maker,Processor,ChipSocket,MBModel,RamVersion,M2-Slot-Number,Price,Stock
==========
1|ASUS|INTEL|1200|ASUS ROG STRIX B560-A GAMING WIFI|DDR4|2|260|5
2|MSI|INTEL|1200|MSI Z590-A PRO|DDR4|3|250|0
3|ASUS|INTEL|1200|ASUS PRIME Z590-P|DDR4|3|270|7
4|ASUS|INTEL|1200|ASUS PRIME Z590-A|DDR4|3|285|0
5|ASUS|INTEL|1200|ASUS ROG STRIX B560-E GAMING WIFI|DDR4|3|300|5
6|MSI|INTEL|1200|MSI MAG Z590 TOMAHAWK WIFI|DDR4|3|320|4
7|GIGABYTE|INTEL|1200|Z590I AORUS ULTRA rev.1.0|DDR4|3|320|4
8|ASUS|INTEL|1200|ASUS ROG STRIX Z590-F GAMING|DDR4|3|350|5
9|GIGABYTE|INTEL|1200|Z590 AORUS ULTRA rev.1.0|DDR4|3|400|3
10|ASUS|INTEL|1700|ASUS PRIME Z690-P D4|DDR4|2|250|6
11|MSI|INTEL|1700|MSI PRO Z690-A|DDR5|4|270|8
12|ASUS|INTEL|1700|ASUS PROART B660-CREATOR D4|DDR4|3|260|7
13|MSI|INTEL|1700|MSI B660 TOMAHAWK WIFI DDR4|DDR4|2|275|5
14|GIGABYTE|INTEL|1700|GIGABYTE Z690 AORUS ELITE DDR4 rev. 1.x|DDR4|3|300|6

15|ASUS|INTEL|1700|ASUS TUF GAMING Z690-PLUS WIFI D4|DDR4|4|360|5
16|GIGABYTE|INTEL|1700|GIGABYTE Z690 Aorus PRO rev. 1.x|DDR5|3|400|6
17|ASUS|INTEL|1700|ASUS ROG STRIX Z690-A GAMING WIFI|DDR5|3|420|5
18|ASUS|INTEL|1700|ASUS ROG STRIX Z690-F GAMING|DDR5|4|500|4
19|ASUS|AMD|AM4|ASUS PRIME X570-P|DDR4|2|190|8
20|ASUS|AMD|AM4|ASUS TUF GAMING B550-PLUS WIFI II|DDR4|2|210|9
21|GIGABYTE|AMD|AM4|GIGABYTE B550 AORUS ELITE AX rev. 1.0|DDR4|2|220|7
22|MSI|AMD|AM4|MSI MAG B550 TOMAHAWK|DDR4|2|235|8
23|ASUS|AMD|AM4|ASUS AMD MB PRIME X570-PRO AM4|DDR4|2|290|6
24|GIGABYTE|AMD|AM4|GIGABYTE X570S AERO G rev. 1.0|DDR4|3|350|5
25|ASUS|AMD|AM5|ASUS TUF GAMING B650-PLUS|DDR5|3|300|4
26|GIGABYTE|AMD|AM5|GIGABYTE B650 AORUS ELITE AX rev. 1.0|DDR5|3|400|3
27|ASUS|AMD|AM5|ASUS TUF GAMING X670E-PLUS WIFI|DDR5|3|420|4
28|ASUS|AMD|AM5|ASUS ROG STRIX B650E-E GAMING WIFI|DDR5|3|450|3
29|ASUS|AMD|AM5|ASUS ROG STRIX X670E-E GAMING WIFI|DDR5|4|600|3
30|ASUS|AMD|AM5|ASUS ROG CROSSHAIR X670E GENE|DDR5|4|700|2
==========
>MarketID=2
==========
Motherboard|ID,Maker,Processor,ChipSocket,MBModel,RamVersion,M2-Slot-Number,Price,Stock
==========
1|ASUS|INTEL|1200|ASUS ROG STRIX B560-A GAMING WIFI|DDR4|2|250|5
2|MSI|INTEL|1200|MSI Z590-A PRO|DDR4|3|260|6
3|ASUS|INTEL|1200|ASUS PRIME Z590-P|DDR4|3|260|0
4|ASUS|INTEL|1200|ASUS PRIME Z590-A|DDR4|3|295|6
5|ASUS|INTEL|1200|ASUS ROG STRIX B560-E GAMING WIFI|DDR4|3|290|0
6|MSI|INTEL|1200|MSI MAG Z590 TOMAHAWK WIFI|DDR4|3|330|4
7|GIGABYTE|INTEL|1200|Z590I AORUS ULTRA rev.1.0|DDR4|3|330|4
8|ASUS|INTEL|1200|ASUS ROG STRIX Z590-F GAMING|DDR4|3|340|5
9|GIGABYTE|INTEL|1200|Z590 AORUS ULTRA rev.1.0|DDR4|3|410|3
10|ASUS|INTEL|1700|ASUS PRIME Z690-P D4|DDR4|2|240|6
11|MSI|INTEL|1700|MSI PRO Z690-A|DDR5|4|280|0
12|ASUS|INTEL|1700|ASUS PROART B660-CREATOR D4|DDR4|3|250|0
13|MSI|INTEL|1700|MSI B660 TOMAHAWK WIFI DDR4|DDR4|2|260|5
14|GIGABYTE|INTEL|1700|GIGABYTE Z690 AORUS ELITE DDR4 rev. 1.x|DDR4|3|315|6
15|ASUS|INTEL|1700|ASUS TUF GAMING Z690-PLUS WIFI D4|DDR4|4|350|5
16|GIGABYTE|INTEL|1700|GIGABYTE Z690 Aorus PRO rev. 1.x|DDR5|3|415|6
17|ASUS|INTEL|1700|ASUS ROG STRIX Z690-A GAMING WIFI|DDR5|3|410|5
18|ASUS|INTEL|1700|ASUS ROG STRIX Z690-F GAMING|DDR5|4|510|4
19|ASUS|AMD|AM4|ASUS PRIME X570-P|DDR4|2|205|0
20|ASUS|AMD|AM4|ASUS TUF GAMING B550-PLUS WIFI II|DDR4|2|200|9
21|GIGABYTE|AMD|AM4|GIGABYTE B550 AORUS ELITE AX rev. 1.0|DDR4|2|230|7
22|MSI|AMD|AM4|MSI MAG B550 TOMAHAWK|DDR4|2|220|8
23|ASUS|AMD|AM4|ASUS AMD MB PRIME X570-PRO AM4|DDR4|2|235|0
24|GIGABYTE|AMD|AM4|GIGABYTE X570S AERO G rev. 1.0|DDR4|3|340|5
25|ASUS|AMD|AM5|ASUS TUF GAMING B650-PLUS|DDR5|3|310|4
26|GIGABYTE|AMD|AM5|GIGABYTE B650 AORUS ELITE AX rev. 1.0|DDR5|3|410|3
27|ASUS|AMD|AM5|ASUS TUF GAMING X670E-PLUS WIFI|DDR5|3|430|0
28|ASUS|AMD|AM5|ASUS ROG STRIX B650E-E GAMING WIFI|DDR5|3|440|0
29|ASUS|AMD|AM5|ASUS ROG STRIX X670E-E GAMING WIFI|DDR5|4|585|3
30|ASUS|AMD|AM5|ASUS ROG CROSSHAIR X670E GENE|DDR5|4|685|2
==========
>MarketID=3
==========
Motherboard|ID,Maker,Processor,ChipSocket,MBModel,RamVersion,M2-Slot-Number,Price,Stock
==========
1|ASUS|INTEL|1200|ASUS ROG STRIX B560-A GAMING WIFI|DDR4|2|270|5
2|MSI|INTEL|1200|MSI Z590-A PRO|DDR4|3|240|6
3|ASUS|INTEL|1200|ASUS PRIME Z590-P|DDR4|3|280|7
4|ASUS|INTEL|1200|ASUS PRIME Z590-A|DDR4|3|280|0
5|ASUS|INTEL|1200|ASUS ROG STRIX B560-E GAMING WIFI|DDR4|3|310|5
6|MSI|INTEL|1200|MSI MAG Z590 TOMAHAWK WIFI|DDR4|3|310|0
7|GIGABYTE|INTEL|1200|Z590I AORUS ULTRA rev.1.0|DDR4|3|310|4
8|ASUS|INTEL|1200|ASUS ROG STRIX Z590-F GAMING|DDR4|3|360|5
9|GIGABYTE|INTEL|1200|Z590 AORUS ULTRA rev.1.0|DDR4|3|390|3
10|ASUS|INTEL|1700|ASUS PRIME Z690-P D4|DDR4|2|265|6
11|MSI|INTEL|1700|MSI PRO Z690-A|DDR5|4|260|8
12|ASUS|INTEL|1700|ASUS PROART B660-CREATOR D4|DDR4|3|270|7
13|MSI|INTEL|1700|MSI B660 TOMAHAWK WIFI DDR4|DDR4|2|285|5
14|GIGABYTE|INTEL|1700|GIGABYTE Z690 AORUS ELITE DDR4 rev. 1.x|DDR4|3|285|6
15|ASUS|INTEL|1700|ASUS TUF GAMING Z690-PLUS WIFI D4|DDR4|4|370|5

16|GIGABYTE|INTEL|1700|GIGABYTE Z690 Aorus PRO rev. 1.x|DDR5|3|385|6
17|ASUS|INTEL|1700|ASUS ROG STRIX Z690-A GAMING WIFI|DDR5|3|430|5
18|ASUS|INTEL|1700|ASUS ROG STRIX Z690-F GAMING|DDR5|4|490|4
19|ASUS|AMD|AM4|ASUS PRIME X570-P|DDR4|2|180|8
20|ASUS|AMD|AM4|ASUS TUF GAMING B550-PLUS WIFI II|DDR4|2|220|9
21|GIGABYTE|AMD|AM4|GIGABYTE B550 AORUS ELITE AX rev. 1.0|DDR4|2|210|0
22|MSI|AMD|AM4|MSI MAG B550 TOMAHAWK|DDR4|2|250|8
23|ASUS|AMD|AM4|ASUS AMD MB PRIME X570-PRO AM4|DDR4|2|280|6
24|GIGABYTE|AMD|AM4|GIGABYTE X570S AERO G rev. 1.0|DDR4|3|365|5
25|ASUS|AMD|AM5|ASUS TUF GAMING B650-PLUS|DDR5|3|310|4
26|GIGABYTE|AMD|AM5|GIGABYTE B650 AORUS ELITE AX rev. 1.0|DDR5|3|410|3
27|ASUS|AMD|AM5|ASUS TUF GAMING X670E-PLUS WIFI|DDR5|3|410|4
28|ASUS|AMD|AM5|ASUS ROG STRIX B650E-E GAMING WIFI|DDR5|3|440|3
29|ASUS|AMD|AM5|ASUS ROG STRIX X670E-E GAMING WIFI|DDR5|4|585|0
30|ASUS|AMD|AM5|ASUS ROG CROSSHAIR X670E GENE|DDR5|4|710|0
==========
>MarketID=1
==========
GPU|ID,Manufacturer,Maker,Model,RAM,Price,Stock
==========
1|ASUS|AMD|ASUS AMD Radeon RX 6600 8GB DUAL-RX6600-8G|8|400|5
2|SAPPHIRE|AMD|SAPPHIRE PCIE SAPPHIRE Pulse RX6600 XT OC 8G 11309-03-20G|8|550|6
3|POWER COLOR|AMD|POWER COLOR Fighter AXRX 6700 10GBD6-3DH/OC AMD|10|500|4
4|GIGABYTE|AMD|GIGABYTE AMD Radeon RX 6700 XT EAGLE 12GB 192bit GV-R67XTEAGLE-12GD|12|580|4
5|ASUS|AMD|ASUS AMD Radeon RX RX6800 16GB 256bit TUF-RX6800-O16G-GAMING|16|880|7
6|SAPPHIRE|AMD|SAPPHIRE AMD Radeon RX 6800 XT 16GB 256bit PULSE GAMING OC RX 6800 XT 16GB|16|1000|6
7|ASUS|AMD|ASUS AMD Radeon RX 6900 XT 16GB 256bit ROG-STRIX-LC-RX6900XT-T16G-GAMING|16|2500|8
8|ASUS|NVIDIA|ASUS NVD RTX 2060 6GB DDR6 192bit DUAL-RTX2060-O6G-EVO|6|500|5
9|GIGABYTE|NVIDIA|GIGABYTE NVidia GeForce RTX 3050 GAMING 8GB 128bit GV-N3050GAMING OC-8GD|8|450|4
10|ASUS|NVIDIA|ASUS NVidia GeForce RTX 3060 12GB 192bit DUAL-RTX3060-O12G-V2 LHR|12|500|6
11|GIGABYTE|NVIDIA|GIGABYTE NVidia GeForce RTX 3060 GAMING OC 12GB 192bit GV-N3060GAMING OC-12G LHR|12|600|5
12|ASUS|NVIDIA|ASUS NVidia GeForce RTX 3070 8GB 256bit DUAL-RTX3070-O8G-V2|8|800|3
13|MSI|NVIDIA|MSI NVidia GeForce RTX 3080 TI 12GB 384bit RTX 3080 Ti VENTUS 3X 12G OC LHR|12|2000|3
14|Inno3D|NVIDIA|Inno3D GeForce RTX 3090 X3 24GB GDDR6X, 384-bit|24|2500|3
==========
>MarketID=2
==========
GPU|ID,Manufacturer,Maker,Model,RAM,Price,Stock
==========
1|ASUS|AMD|ASUS AMD Radeon RX 6600 8GB DUAL-RX6600-8G|8|410|5
2|SAPPHIRE|AMD|SAPPHIRE PCIE SAPPHIRE Pulse RX6600 XT OC 8G 11309-03-20G|8|540|5
3|POWER COLOR|AMD|POWER COLOR Fighter AXRX 6700 10GBD6-3DH/OC AMD|10|510|5
4|GIGABYTE|AMD|GIGABYTE AMD Radeon RX 6700 XT EAGLE 12GB 192bit GV-R67XTEAGLE-12GD|12|570|0
5|ASUS|AMD|ASUS AMD Radeon RX RX6800 16GB 256bit TUF-RX6800-O16G-GAMING|16|890|5
6|SAPPHIRE|AMD|SAPPHIRE AMD Radeon RX 6800 XT 16GB 256bit PULSE GAMING OC RX 6800 XT 16GB|16|985|5
7|ASUS|AMD|ASUS AMD Radeon RX 6900 XT 16GB 256bit ROG-STRIX-LC-RX6900XT-T16G-GAMING|16|2540|0
8|ASUS|NVIDIA|ASUS NVD RTX 2060 6GB DDR6 192bit DUAL-RTX2060-O6G-EVO|6|485|5
9|GIGABYTE|NVIDIA|GIGABYTE NVidia GeForce RTX 3050 GAMING 8GB 128bit GV-N3050GAMING OC-8GD|8|440|5
10|ASUS|NVIDIA|ASUS NVidia GeForce RTX 3060 12GB 192bit DUAL-RTX3060-O12G-V2 LHR|12|0|5
11|GIGABYTE|NVIDIA|GIGABYTE NVidia GeForce RTX 3060 GAMING OC 12GB 192bit GV-N3060GAMING OC-12G LHR|12|620|5
12|ASUS|NVIDIA|ASUS NVidia GeForce RTX 3070 8GB 256bit DUAL-RTX3070-O8G-V2|8|790|5
13|MSI|NVIDIA|MSI NVidia GeForce RTX 3080 TI 12GB 384bit RTX 3080 Ti VENTUS 3X 12G OC LHR|12|2050|5
14|Inno3D|NVIDIA|Inno3D GeForce RTX 3090 X3 24GB GDDR6X, 384-bit|24|2480|5
==========
>MarketID=3
==========
GPU|ID,Manufacturer,Maker,Model,RAM,Price,Stock
==========
1|ASUS|AMD|ASUS AMD Radeon RX 6600 8GB DUAL-RX6600-8G|8|395|7
2|SAPPHIRE|AMD|SAPPHIRE PCIE SAPPHIRE Pulse RX6600 XT OC 8G 11309-03-20G|8|560|8
3|POWER COLOR|AMD|POWER COLOR Fighter AXRX 6700 10GBD6-3DH/OC AMD|10|490|9
4|GIGABYTE|AMD|GIGABYTE AMD Radeon RX 6700 XT EAGLE 12GB 192bit GV-R67XTEAGLE-12GD|12|590|0
5|ASUS|AMD|ASUS AMD Radeon RX RX6800 16GB 256bit TUF-RX6800-O16G-GAMING|16|870|4
6|SAPPHIRE|AMD|SAPPHIRE AMD Radeon RX 6800 XT 16GB 256bit PULSE GAMING OC RX 6800 XT 16GB|16|1020|3
7|ASUS|AMD|ASUS AMD Radeon RX 6900 XT 16GB 256bit ROG-STRIX-LC-RX6900XT-T16G-GAMING|16|2480|5
8|ASUS|NVIDIA|ASUS NVD RTX 2060 6GB DDR6 192bit DUAL-RTX2060-O6G-EVO|6|505|4
9|GIGABYTE|NVIDIA|GIGABYTE NVidia GeForce RTX 3050 GAMING 8GB 128bit GV-N3050GAMING OC-8GD|8|435|5
10|ASUS|NVIDIA|ASUS NVidia GeForce RTX 3060 12GB 192bit DUAL-RTX3060-O12G-V2 LHR|12|490|4
11|GIGABYTE|NVIDIA|GIGABYTE NVidia GeForce RTX 3060 GAMING OC 12GB 192bit GV-N3060GAMING OC-12G LHR|12|0|6
12|ASUS|NVIDIA|ASUS NVidia GeForce RTX 3070 8GB 256bit DUAL-RTX3070-O8G-V2|8|810|5
13|MSI|NVIDIA|MSI NVidia GeForce RTX 3080 TI 12GB 384bit RTX 3080 Ti VENTUS 3X 12G OC LHR|12|1980|4

14|Inno3D|NVIDIA|Inno3D GeForce RTX 3090 X3 24GB GDDR6X, 384-bit|24|2540|0
==========
>MarketID=1
==========
RAM|ID,Manufacturer,Model,RAMVersion,Capacity,MHZ,CL,Price,Stock
==========
1|KINGSTON|KINGSTON DIMM DDR4 16GB 3200MHz KF432C16RB1A/16 Fury Renegade RGB|DDR4|16|3200|16|80|4
2|KINGSTON|KINGSTON DIMM DDR4 32GB 3200MHz KF432C16RB/32 Fury Renegade Black|DDR4|32|3200|16|165|0
3|KINGSTON|KINGSTON DIMM DDR4 64GB (2x32GB kit) 3200MHz KF432C16BBK2/64 Fury Beast Black|DDR4|64|3200|16|260|6
4|KINGSTON|KINGSTON DIMM DDR4 16GB 3600MHz KF436C16RB1A/16 Fury Renegade RGB|DDR4|16|3600|16|105|5
5|KINGSTON|KINGSTON DIMM DDR4 32GB (2x16GB) 3600MHz KF436C16RB1AK2/32 Fury Renegade RGB|DDR4|32|3600|16|190|7
6|KINGSTON|KINGSTON RAM DDR4 32GB (2x16GB) 3600MHz Kingston Fury Beast Black KF436C18BBK2/32|DDR4|32|3600|18|155|5
7|KINGSTON|KINGSTON DIMM DDR4 64GB (2x32GB kit) 3600MHz KF436C18RBK2/64 Fury Renegade Black|DDR4|64|3600|18|330|4
8|KINGSTON|KINGSTON DIMM DDR5 16GB 4800MT/s KF548C38BBA-16 Fury Beast RGB|DDR5|16|4800|38|100|0
9|KINGSTON|ADATA DDR5 32GB 4800MHz (2x16) AData AD5U480016G-DT|DDR5|32|4800|38|185|0
10|KINGSTON|KINGSTON DIMM DDR5 32GB (2x16GB kit) 6000MT/s KF560C32RSK2-32 FURY Renegade|DDR5|32|6000|32|330|8
11|KINGSTON|KINGSTON DIMM DDR5 16GB 6400MT/s KF564C32RSA-16 FURY Renegade RGB|DDR5|16|6400|32|230|6
12|KINGSTON|KINGSTON DIMM DDR5 32GB (2x16GB kit) 6400MT/s KF564C32RSK2-32 FURY Renegade|DDR5|32|6400|32|390|0
==========
>MarketID=2
==========
RAM|ID,Manufacturer,Model,RAMVersion,Capacity,MHZ,CL,Price,Stock
==========
1|KINGSTON|KINGSTON DIMM DDR4 16GB 3200MHz KF432C16RB1A/16 Fury Renegade RGB|DDR4|16|3200|16|90|10
2|KINGSTON|KINGSTON DIMM DDR4 32GB 3200MHz KF432C16RB/32 Fury Renegade Black|DDR4|32|3200|16|155|8
3|KINGSTON|KINGSTON DIMM DDR4 64GB (2x32GB kit) 3200MHz KF432C16BBK2/64 Fury Beast Black|DDR4|64|3200|16|270|6
4|KINGSTON|KINGSTON DIMM DDR4 16GB 3600MHz KF436C16RB1A/16 Fury Renegade RGB|DDR4|16|3600|16|95|9
5|KINGSTON|KINGSTON DIMM DDR4 32GB (2x16GB) 3600MHz KF436C16RB1AK2/32 Fury Renegade RGB|DDR4|32|3600|16|180|7
6|KINGSTON|KINGSTON RAM DDR4 32GB (2x16GB) 3600MHz Kingston Fury Beast Black KF436C18BBK2/32|DDR4|32|3600|18|165|5
7|KINGSTON|KINGSTON DIMM DDR4 64GB (2x32GB kit) 3600MHz KF436C18RBK2/64 Fury Renegade Black|DDR4|64|3600|18|320|8
8|KINGSTON|KINGSTON DIMM DDR5 16GB 4800MT/s KF548C38BBA-16 Fury Beast RGB|DDR5|16|4800|38|110|7
9|KINGSTON|ADATA DDR5 32GB 4800MHz (2x16) AData AD5U480016G-DT|DDR5|32|4800|38|195|6
10|KINGSTON|KINGSTON DIMM DDR5 32GB (2x16GB kit) 6000MT/s KF560C32RSK2-32 FURY Renegade|DDR5|32|6000|32|320|6
11|KINGSTON|KINGSTON DIMM DDR5 16GB 6400MT/s KF564C32RSA-16 FURY Renegade RGB|DDR5|16|6400|32|220|4
12|KINGSTON|KINGSTON DIMM DDR5 32GB (2x16GB kit) 6400MT/s KF564C32RSK2-32 FURY Renegade|DDR5|32|6400|32|400|4
==========
>MarketID=3
==========
RAM|ID,Manufacturer,Model,RAMVersion,Capacity,MHZ,CL,Price,Stock
==========
1|KINGSTON|KINGSTON DIMM DDR4 16GB 3200MHz KF432C16RB1A/16 Fury Renegade RGB|DDR4|16|3200|16|100|8
2|KINGSTON|KINGSTON DIMM DDR4 32GB 3200MHz KF432C16RB/32 Fury Renegade Black|DDR4|32|3200|16|150|6
3|KINGSTON|KINGSTON DIMM DDR4 64GB (2x32GB kit) 3200MHz KF432C16BBK2/64 Fury Beast Black|DDR4|64|3200|16|280|0
4|KINGSTON|KINGSTON DIMM DDR4 16GB 3600MHz KF436C16RB1A/16 Fury Renegade RGB|DDR4|16|3600|16|90|4
5|KINGSTON|KINGSTON DIMM DDR4 32GB (2x16GB) 3600MHz KF436C16RB1AK2/32 Fury Renegade RGB|DDR4|32|3600|16|170|8
6|KINGSTON|KINGSTON RAM DDR4 32GB (2x16GB) 3600MHz Kingston Fury Beast Black KF436C18BBK2/32|DDR4|32|3600|18|180|6
7|KINGSTON|KINGSTON DIMM DDR4 64GB (2x32GB kit) 3600MHz KF436C18RBK2/64 Fury Renegade Black|DDR4|64|3600|18|310|0
8|KINGSTON|KINGSTON DIMM DDR5 16GB 4800MT/s KF548C38BBA-16 Fury Beast RGB|DDR5|16|4800|38|120|6
9|KINGSTON|ADATA DDR5 32GB 4800MHz (2x16) AData AD5U480016G-DT|DDR5|32|4800|38|200|6
10|KINGSTON|KINGSTON DIMM DDR5 32GB (2x16GB kit) 6000MT/s KF560C32RSK2-32 FURY Renegade|DDR5|32|6000|32|310|6
11|KINGSTON|KINGSTON DIMM DDR5 16GB 6400MT/s KF564C32RSA-16 FURY Renegade RGB|DDR5|16|6400|32|210|0
12|KINGSTON|KINGSTON DIMM DDR5 32GB (2x16GB kit) 6400MT/s KF564C32RSK2-32 FURY Renegade|DDR5|32|6400|32|405|4
==========
>MarketID=1
==========
PowerSupply|ID,Manufacturer,Model,Capacity,Price,Stock
==========
1|CHIEFTEC|CHIEFTEC GPS-600A8 600W Full Smart|600|50|0
2|THERMALTAKE|THERMALTAKE Smart RGB 600W, PS-SPR-0600NHSAWE-1|600|80|10
3|COOLER MASTER|COOLER MASTER MWE White 600W|600|70|10
4|COOLER MASTER|COOLER MASTER MWE Bronze V2 650W|650|125|8
5|NZXT|NZXT C650 650W|650|135|0
6|ASUS|ASUS ROG-STRIX-650G 650W|650|130|8
7|CHIEFTEC|CHIEFTEC PPS-750FC 750W|750|145|0
8|NZXT|NZXT C750 750W|750|160|0
9|COOLER MASTER|COOLER MASTER MWE Gold V2 850W|850|140|12
10|CHIEFTEC|CHIEFTEC PPS-850FC 850W|850|170|10
11|BE QUIET|BE QUIET STRAIGHT POWER 11 PLATINUM 1000W, 80 PLUS Platinum|1000|250|0
12|CHIEFTEC|CHIEFTEC PPS-1250FC 1250W|1250|295|6
==========
>MarketID=2

```
==========================================================================================================================
PowerSupply|ID,Manufacturer,Model,Capacity,Price,Stock
==========================================================================================================================
1|CHIEFTEC|CHIEFTEC GPS-600A8 600W Full Smart|600|70|10
2|THERMALTAKE|THERMALTAKE Smart RGB 600W, PS-SPR-0600NHSAWE-1|600|50|0
3|COOLER MASTER|COOLER MASTER MWE White 600W|600|90|10
4|COOLER MASTER|COOLER MASTER MWE Bronze V2 650W|650|105|0
5|NZXT|NZXT C650 650W|650|115|9
6|ASUS|ASUS ROG-STRIX-650G 650W|650|150|8
7|CHIEFTEC|CHIEFTEC PPS-750FC 750W|750|125|12
8|NZXT|NZXT C750 750W|750|180|12
9|COOLER MASTER|COOLER MASTER MWE Gold V2 850W|850|155|12
10|CHIEFTEC|CHIEFTEC PPS-850FC 850W|850|185|10
11|BE QUIET|BE QUIET STRAIGHT POWER 11 PLATINUM 1000W, 80 PLUS Platinum|1000|230|8
12|CHIEFTEC|CHIEFTEC PPS-1250FC 1250W|1250|275|0
==========================================================================================================================
>MarketID=3
==========================================================================================================================
PowerSupply|ID,Manufacturer,Model,Capacity,Price,Stock
==========================================================================================================================
1|CHIEFTEC|CHIEFTEC GPS-600A8 600W Full Smart|600|60|10
2|THERMALTAKE|THERMALTAKE Smart RGB 600W, PS-SPR-0600NHSAWE-1|600|70|10
3|COOLER MASTER|COOLER MASTER MWE White 600W|600|80|10
4|COOLER MASTER|COOLER MASTER MWE Bronze V2 650W|650|115|8
5|NZXT|NZXT C650 650W|650|125|9
6|ASUS|ASUS ROG-STRIX-650G 650W|650|140|8
7|CHIEFTEC|CHIEFTEC PPS-750FC 750W|750|135|12
8|NZXT|NZXT C750 750W|750|170|12
9|COOLER MASTER|COOLER MASTER MWE Gold V2 850W|850|150|12
10|CHIEFTEC|CHIEFTEC PPS-850FC 850W|850|180|10
11|BE QUIET|BE QUIET STRAIGHT POWER 11 PLATINUM 1000W, 80 PLUS Platinum|1000|240|8
12|CHIEFTEC|CHIEFTEC PPS-1250FC 1250W|1250|285|6
==========================================================================================================================
>MarketID=1
==========================================================================================================================
PcCase|ID,Manufacturer,Model,FrontPanel,TopPanel,Price,Stock
==========================================================================================================================
1|DEEPCOOL|DEEPCOOL Kućište CL500 4F|3x120mm|2x120mm|150|10
2|NZXT|NZXT H7 Black|3x140mm-3x120mm|2x120mm-2x140mm|180|0
3|CHIEFTEC|CHIEFTEC Kućište GP-03B-OP|1x180mm-3x120mm|2x140mm-2x120mm|170|10
4|CHIEFTEC|CHIEFTEC GB-AC300G|2x140mm-3x120mm|2x120mm-2x140mm|220|0
5|CORSAIR|CORSAIR Mid-Tower Gaming Carbide Spec-Omega RGB|2x140mm|2x120mm-3x120mm|225|0
6|NZXT|NZXT H7 Elite|3x140mm-3x120mm|3x120mm-3x140mm|230|10
7|COOLER MASTER|COOLER MASTER MasterCase H500P Mesh ARGB|2x200mm|3x120mm-3x140mm|260|10
8|CORSAIR|CORSAIR Mid-Tower Gaming, Obsidian 500D Premium|2x140mm|2x120mm-3x120mm|260|10
9|ASUS|ASUS GX601 ROG STRIX HELIOS|2x140mm-3x120mm|2x140mm-3x120mm|340|10
10|COOLER MASTER|COOLER MASTER Cosmos C700P|3x140mm-3x120mm|3x140mm-3x120mm|390|10
==========================================================================================================================
>MarketID=2
==========================================================================================================================
PcCase|ID,Manufacturer,Model,FrontPanel,TopPanel,Price,Stock
==========================================================================================================================
1|DEEPCOOL|DEEPCOOL Kućište CL500 4F|3x120mm|2x120mm|160|10
2|NZXT|NZXT H7 Black|3x140mm-3x120mm|2x120mm-2x140mm|170|10
3|CHIEFTEC|CHIEFTEC Kućište GP-03B-OP|1x180mm-3x120mm|2x140mm-2x120mm|180|10
4|CHIEFTEC|CHIEFTEC GB-AC300G|2x140mm-3x120mm|2x120mm-2x140mm|210|10
5|CORSAIR|CORSAIR Mid-Tower Gaming Carbide Spec-Omega RGB|2x140mm|2x120mm-3x120mm|215|10
6|NZXT|NZXT H7 Elite|3x140mm-3x120mm|3x120mm-3x140mm|240|10
7|COOLER MASTER|COOLER MASTER MasterCase H500P Mesh ARGB|2x200mm|3x120mm-3x140mm|250|10
8|CORSAIR|CORSAIR Mid-Tower Gaming, Obsidian 500D Premium|2x140mm|2x120mm-3x120mm|270|10
9|ASUS|ASUS GX601 ROG STRIX HELIOS|2x140mm-3x120mm|2x140mm-3x120mm|350|10
10|COOLER MASTER|COOLER MASTER Cosmos C700P|3x140mm-3x120mm|3x140mm-3x120mm|400|10
==========================================================================================================================
>MarketID=3
==========================================================================================================================
PcCase|ID,Manufacturer,Model,FrontPanel,TopPanel,Price,Stock
==========================================================================================================================
1|DEEPCOOL|DEEPCOOL Kućište CL500 4F|3x120mm|2x120mm|180|10
2|NZXT|NZXT H7 Black|3x140mm-3x120mm|2x120mm-2x140mm|160|0
3|CHIEFTEC|CHIEFTEC Kućište GP-03B-OP|1x180mm-3x120mm|2x140mm-2x120mm|190|0
4|CHIEFTEC|CHIEFTEC GB-AC300G|2x140mm-3x120mm|2x120mm-2x140mm|200|10
```

5|CORSAIR|CORSAIR Mid-Tower Gaming Carbide Spec-Omega RGB|2x140mm|2x120mm-3x120mm|205|10
6|NZXT|NZXT H7 Elite|3x140mm-3x120mm|3x120mm-3x140mm|250|10
7|COOLER MASTER|COOLER MASTER MasterCase H500P Mesh ARGB|2x200mm|3x120mm-3x140mm|240|0
8|CORSAIR|CORSAIR Mid-Tower Gaming, Obsidian 500D Premium|2x140mm|2x120mm-3x120mm|260|10
9|ASUS|ASUS GX601 ROG STRIX HELIOS|2x140mm-3x120mm|2x140mm-3x120mm|355|10
10|COOLER MASTER|COOLER MASTER Cosmos C700P|3x140mm-3x120mm|3x140mm-3x120mm|405|10
==========
>MarketID=1
==========
Cooler|ID,Manufacturer,Model,Size,SocketSupport,Price,Stock
==========
1|COOLER MASTER|COOLER MASTER MasterLiquid ML240L V2 RGB|2x120mm|1200-AM4|110|10
2|BE QUIET|BE QUIET CPU Pure Loop 360mm BW008|2x120mm|1200-AM4|130|10
3|NZXT|NZXT Kraken X63|2x120mm|1200-1700-AM4|130|10
4|BE QUIET|BE QUIET CPU Cooler Be quiet Pure Loop 2 FX 240mm BW013|3x120mm|1200-1700-AM4-AM5|160|10
5|COOLER MASTER|COOLER MASTER MasterLiquid PL240 FLUX|2x120mm|1200-AM4|180|10
6|GIGABYTE|GIGABYTE AORUS WATERFORCE 280|2x120mm|1200-1700-AM4|200|10
7|CORSAIR|CORSAIR ICUE H150i ELITE CAPELLIX|3x120mm|1200-1700-AM4-AM5|240|10
8|NZXT|NZXT Kraken Z63|2x120mm|1200-1700-AM4-AM5|250|10
9|NZXT|NZXT Kraken Z73|3x120mm|1200-1700-AM4-AM5|300|10
==========
>MarketID=2
==========
Cooler|ID,Manufacturer,Model,Size,SocketSupport,Price,Stock
==========
1|COOLER MASTER|COOLER MASTER MasterLiquid ML240L V2 RGB|2x120mm|1200-AM4|100|10
2|BE QUIET|BE QUIET CPU Pure Loop 360mm BW008|2x120mm|1200-AM4|140|0
3|NZXT|NZXT Kraken X63|2x120mm|1200-1700-AM4|140|10
4|BE QUIET|BE QUIET CPU Cooler Be quiet Pure Loop 2 FX 240mm BW013|3x120mm|1200-1700-AM4-AM5|150|10
5|COOLER MASTER|COOLER MASTER MasterLiquid PL240 FLUX|2x120mm|1200-AM4|190|10
6|GIGABYTE|GIGABYTE AORUS WATERFORCE 280|2x120mm|1200-1700-AM4|205|10
7|CORSAIR|CORSAIR ICUE H150i ELITE CAPELLIX|3x120mm|1200-1700-AM4-AM5|230|0
8|NZXT|NZXT Kraken Z63|2x120mm|1200-1700-AM4-AM5|260|10
9|NZXT|NZXT Kraken Z73|3x120mm|1200-1700-AM4-AM5|305|0
==========
>MarketID=3
==========
Cooler|ID,Manufacturer,Model,Size,SocketSupport,Price,Stock
==========
1|COOLER MASTER|COOLER MASTER MasterLiquid ML240L V2 RGB|2x120mm|1200-AM4|100|10
2|BE QUIET|BE QUIET CPU Pure Loop 360mm BW008|2x120mm|1200-AM4|150|10
3|NZXT|NZXT Kraken X63|2x120mm|1200-1700-AM4|130|10
4|BE QUIET|BE QUIET CPU Cooler Be quiet Pure Loop 2 FX 240mm BW013|3x120mm|1200-1700-AM4-AM5|160|0
5|COOLER MASTER|COOLER MASTER MasterLiquid PL240 FLUX|2x120mm|1200-AM4|185|10
6|GIGABYTE|GIGABYTE AORUS WATERFORCE 280|2x120mm|1200-1700-AM4|215|10
7|CORSAIR|CORSAIR ICUE H150i ELITE CAPELLIX|3x120mm|1200-1700-AM4-AM5|225|0
8|NZXT|NZXT Kraken Z63|2x120mm|1200-1700-AM4-AM5|265|0
9|NZXT|NZXT Kraken Z73|3x120mm|1200-1700-AM4-AM5|315|10
==========
>MarketID=1
==========
SSD|ID,Manufacturer,Model,Size,PCISupport,Price,Stock
==========
1|SAMSUNG|SAMSUNG 250GB M.2 NVMe MZ-V7S250BW 970 EVO PLUS Series SSD|250GB|3.0|50|10
2|SAMSUNG|SAMSUNG 500GB M.2 NVMe MZ-V8V500BW 980 Series SSD|500GB|3.0|70|10
3|KINGSTON|KINGSTON 500GB M.2 NVMe SFYRS/500G SSD FURY Renegade|500GB|4.0|80|0
4|SAMSUNG|SAMSUNG 500GB M.2 NVMe MZ-V8P500BW 980 Pro Series SSD|500GB|4.0|105|0
5|SAMSUNG|SAMSUNG 1TB M.2 NVMe MZ-V7S1T0BW 970 EVO PLUS Series SSD|1TB|3.0|130|10
6|SAMSUNG|SAMSUNG 1TB M.2 NVMe MZ-V8P1T0CW 980 Pro Series Heatsink SSD|1TB|4.0|185|10
7|SAMSUNG|SAMSUNG 2TB M.2 NVMe MZ-V7S2T0BW 970 EVO PLUS Series SSD|2TB|3.0|250|10
8|KINGSTON|KINGSTON 2TB M.2 NVMe SKC3000D/2048G SSD KC3000 series|2TB|4.0|385|10
9|SAMSUNG|SAMSUNG 2TB M.2 NVMe MZ-V8P2T0BW 980 Pro Series SSD|2TB|4.0|485|10
10|KINGSTON|KINGSTON 4TB M.2 NVMe SFYRD/4000G SSD FURY Renegade HDD03581|4TB|4.0|490|0
==========
>MarketID=2
==========
SSD|ID,Manufacturer,Model,Size,PCISupport,Price,Stock
==========
1|SAMSUNG|SAMSUNG 250GB M.2 NVMe MZ-V7S250BW 970 EVO PLUS Series SSD|250GB|3.0|65|10
2|SAMSUNG|SAMSUNG 500GB M.2 NVMe MZ-V8V500BW 980 Series SSD|500GB|3.0|70|10
3|KINGSTON|KINGSTON 500GB M.2 NVMe SFYRS/500G SSD FURY Renegade|500GB|4.0|95|10

4|SAMSUNG|SAMSUNG 500GB M.2 NVMe MZ-V8P500BW 980 Pro Series SSD|500GB|4.0|105|0
5|SAMSUNG|SAMSUNG 1TB M.2 NVMe MZ-V7S1T0BW 970 EVO PLUS Series SSD|1TB|3.0|130|0
6|SAMSUNG|SAMSUNG 1TB M.2 NVMe MZ-V8P1T0CW 980 Pro Series Heatsink SSD|1TB|4.0|175|10
7|SAMSUNG|SAMSUNG 2TB M.2 NVMe MZ-V7S2T0BW 970 EVO PLUS Series SSD|2TB|3.0|245|10
8|KINGSTON|KINGSTON 2TB M.2 NVMe SKC3000D/2048G SSD KC3000 series|2TB|4.0|370|10
9|SAMSUNG|SAMSUNG 2TB M.2 NVMe MZ-V8P2T0BW 980 Pro Series SSD|2TB|4.0|505|0
10|KINGSTON|KINGSTON 4TB M.2 NVMe SFYRD/4000G SSD FURY Renegade HDD03581|4TB|4.0|495|10
==========
>MarketID=3
==========
SSD|ID,Manufacturer,Model,Size,SocketSupport,Price,Stock
==========
1|SAMSUNG|SAMSUNG 250GB M.2 NVMe MZ-V7S250BW 970 EVO PLUS Series SSD|250GB|3.0|60|10
2|SAMSUNG|SAMSUNG 500GB M.2 NVMe MZ-V8V500BW 980 Series SSD|500GB|3.0|65|10
3|KINGSTON|KINGSTON 500GB M.2 NVMe SFYRS/500G SSD FURY Renegade|500GB|4.0|90|10
4|SAMSUNG|SAMSUNG 500GB M.2 NVMe MZ-V8P500BW 980 Pro Series SSD|500GB|4.0|100|0
5|SAMSUNG|SAMSUNG 1TB M.2 NVMe MZ-V7S1T0BW 970 EVO PLUS Series SSD|1TB|3.0|135|10
6|SAMSUNG|SAMSUNG 1TB M.2 NVMe MZ-V8P1T0CW 980 Pro Series Heatsink SSD|1TB|4.0|180|10
7|SAMSUNG|SAMSUNG 2TB M.2 NVMe MZ-V7S2T0BW 970 EVO PLUS Series SSD|2TB|3.0|240|10
8|KINGSTON|KINGSTON 2TB M.2 NVMe SKC3000D/2048G SSD KC3000 series|2TB|4.0|380|10
9|SAMSUNG|SAMSUNG 2TB M.2 NVMe MZ-V8P2T0BW 980 Pro Series SSD|2TB|4.0|500|10
10|KINGSTON|KINGSTON 4TB M.2 NVMe SFYRD/4000G SSD FURY Renegade HDD03581|4TB|4.0|500|10
==========
==========

Task Description

==========

We need to create search engine to buy, search and suggest specific components or entire computer configurations.

Alex wants to buy motherboard for 1700 socket in price range between 300 and 400.
Ben want to get info about best price for Intel CPU 12th generation with 12 cores.

Cane wants to get info about difference between CPU:
Intel 12th generation with 16 cores and AMD 5000 series 16 cores.

Diana wants to get info about difference between AMD 7000 series CPU with 8,12 and 16 cores.

David wants to buy CPU: Intel 12900K,AMD 5950X and AMD 7950X.

Elena wants to get info about all motherboards which are Z690 models in price range between 400 and 450.

Fred wants to buy the cheapest motherboard for AMD with AM4 chip socket. He needs two of them.

Gina wants to buy the most expensive ASUS motherboard which supports AM5 and DDR5 with 4 slots for M2 SSD.

Helen wants to get info about all motherboards for AMD which supports DDR4 with 2 slots for M2 SSD.
Price range should be between 200 and 250.

Kyle wants to buy the most expensive motherboard for Intel i9 12900K CPU.

Paul wants to buy cooler for NZXT H7 PC case, for top panel. Cooler needs to have support for i9 12900K CPU and needs to be in price range between 220 and 250. He needs two of them.

Lara wants to by RAM:
- 4 RAM slots with DDR4 support on 3600MHz with CL-16
- 4 RAM slots with DDR5 support on 6400Mhz with CL-32

Tina wants to buy SSD: KINGSTON 500GB M.2, two of them and SAMSUNG 500GB M.2, also two of them.
If there are both of them available, she wants to take both.
If only one is available, she wants to get that anyway without second item.

Victor wants to buy whole PC configuration with following specifications:

- CPU = AMD 7950X
- GPU = AMD RX6600 XT with 8GB of RAM
- Motherboard = The most expensive MB for that CPU
- RAM = DDR5 6400MHz with CL-32

- PowerSupply = CHIEFTEC 850W in price range between 150 and 200
- PcCase = NZXT with 3x120mm support for front and top panel in price range between 200 and 250
- Cooler = CORSAIR to support 7950X and 3x120mm size
- SSD = Three M2 1TB which support PCI 4.0

Simon will be our delivery support and he want to update markets with new items delivery from factories:

```
==========================================================================================================
>MarketID=1
==========================================================================================================
DeliveryInfo|ItemType,ItemID,NewDelivery
==========================================================================================================
CPU|10|2
CPU|20|3
Motherboard|8|3
Motherboard|16|2
Motherboard|21|1
GPU|5|3
GPU|7|2
GPU|10|2
RAM|8|8
RAM|10|8
RAM|12|8
PowerSupply|7|4
PowerSupply|9|6
PcCase|4|6
PcCase|6|4
Cooler|5|3
Cooler|7|3
SSD|5|4
SSD|9|4
==========================================================================================================
>MarketID=2
==========================================================================================================
DeliveryInfo|ItemType,ItemID,NewDelivery
==========================================================================================================
CPU|9|2
CPU|18|3
Motherboard|7|3
Motherboard|15|2
Motherboard|20|1
GPU|4|3
GPU|6|2
GPU|9|2
RAM|7|8
RAM|9|8
RAM|11|8
PowerSupply|8|6
PowerSupply|10|4
PcCase|3|6
PcCase|5|4
Cooler|4|3
Cooler|6|3
SSD|6|4
SSD|8|4
==========================================================================================================
```

NOTE: You need to create at least single test for each query type in search engine which you execute.