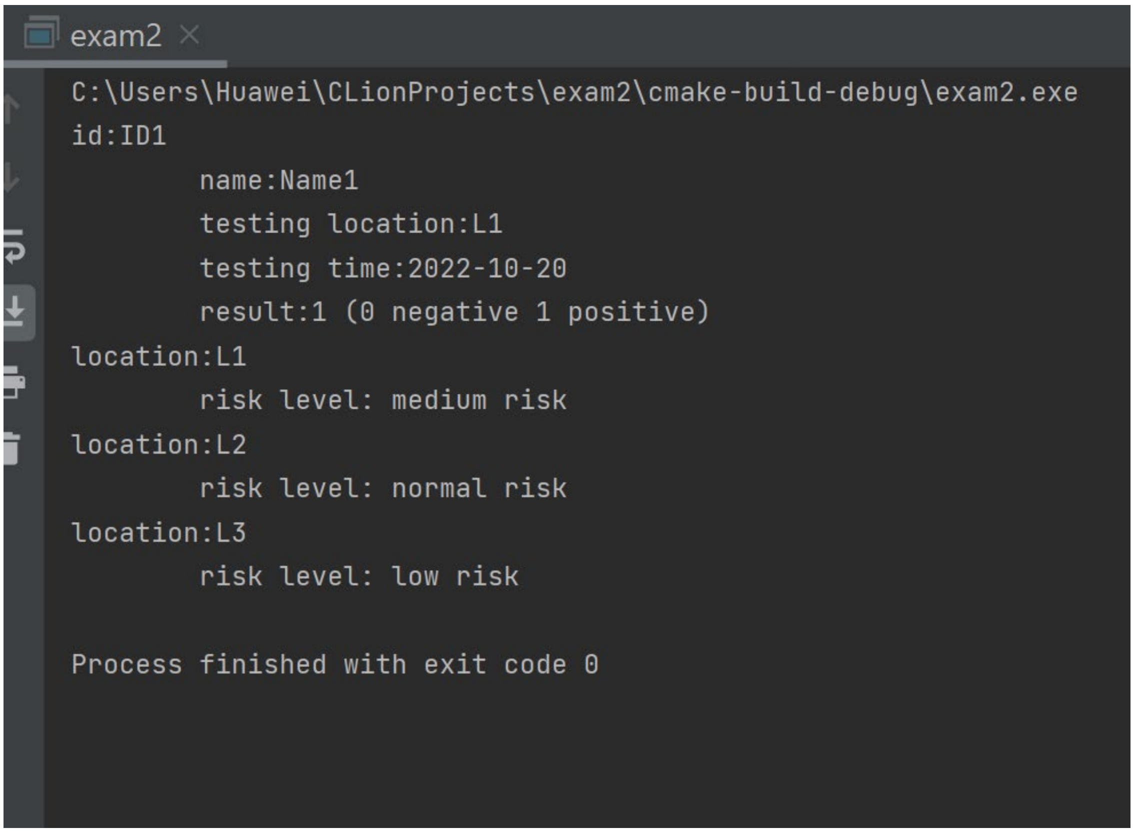
Created by STUDENT ID 2156257 (Gianmarco Fortunelli)

My program can work



```
C:\Users\Huawei\CLionProjects\exam2\cmake-build-debug\exam2.exe
id:ID1
        name:Name1
        testing location:L1
        testing time:2022-10-20
        result:1 (0 negative 1 positive)
location:L1
        risk level: medium risk
location:L2
        risk level: normal risk
location:L3
        risk level: low risk

Process finished with exit code 0
```

1. What information should we record?

   We can divide two different information to record link to the two different output required by the function:
   Firstly, we should record COVID test info in a data structure in the form struct person {ID, name, testing time, testing location, result}
   Them we use another struct location {testing location, number of positive} to define the level of risk of the area

2. What operations the program needs to perform

   Firstly, we make a function of insert all the data except of result; then we use another function to insert the result of test. We need other 2 functions of search about this data. To minimize the time complexity, I should use two hash table. The first hash table contains all the data of the single person while the second one collects the data of each location and determine the level of risk.

   Pseudo code

```
main(){
initHashtablePerson();
initHashtableLocation();

addTestinfo(id, name, location, time);
addTestRes(id, result);
….
….
searchTestingRes(id);
searchRegionRisk(name);

}
```

```
addTestingInfo(char* ID, char* Name, char* testingTime, char* testingLocation){
//add to hash table with flag value
Index=hashFunPerson(id);
Pointer=tab1[index];
Look if Already Present else create a new node;
Insert value;}
```

Function complexity depend on the number of collision present in the table. If there aren't collision the complexity of the function is O(1) else is the number element that have the same key.

```
addTestRes(id, result){
findPerson on the hash table;
analyse result and insert in the person element;
if from positive to negative remove one person number of positive of the location;
else if from negative to positive add one;
}
```

Function complexity depend on the number of collision (all on the search function). As before we get index from hashFunPerson and we compare element with same index until we find the same id; same things also in search in location hashtable. If not collision in both table O(1).

```
searchTestingRed(id){
index=hashFunPerson(id);
check in the chain the correct element;
print data;
}
```

Function Complexity depend on length of the chain in case of collision in hashTable of person. If not collision O(1);

```
searchRegionRisk(loc){
index=hashFunLocation(loc);
check the chain until the correct element;
analyse number of positive and print risk level;
}
```

Function complexity depend on number of collision between name of location. If not collision O(1).

In total the complexity of all the function could be O(1) defining a correct size of the table and correct hash function to avoid collision.