

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 5 REPORT

**İslam Göktan Selçuk
1414044071**

Course Assistant: Fatma Nur Esirci

1 Double Hashing Map

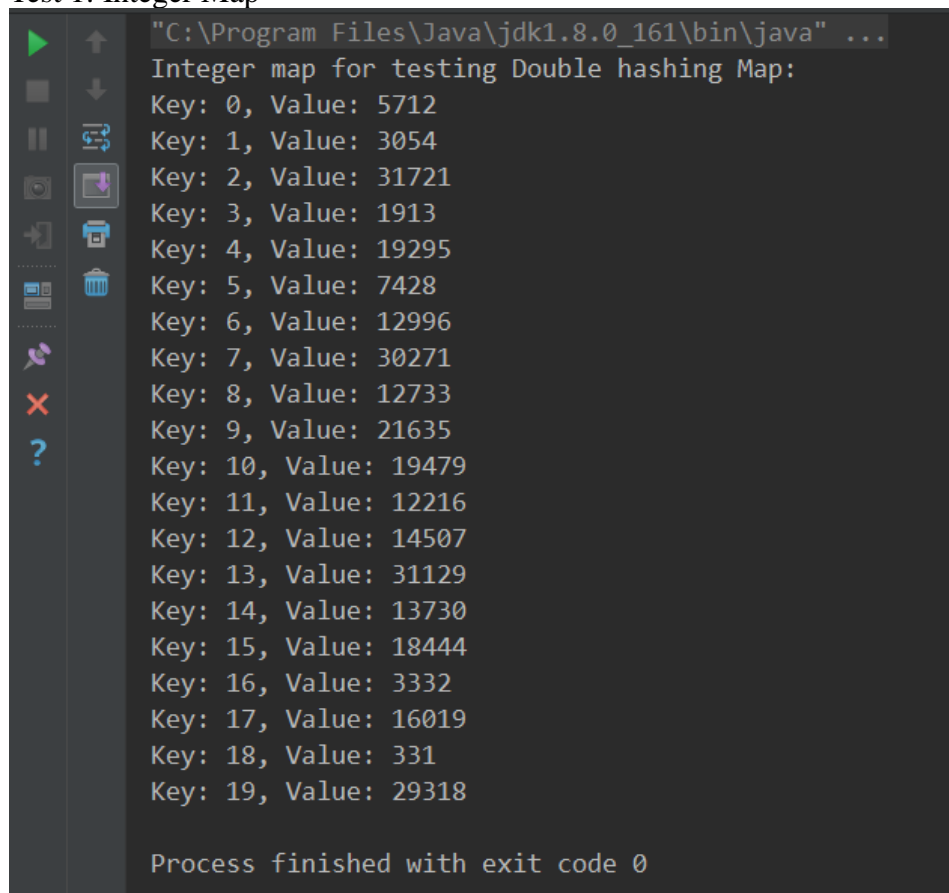
This part about Question1 in HW5

1.1 Pseudocode and Explanation

...

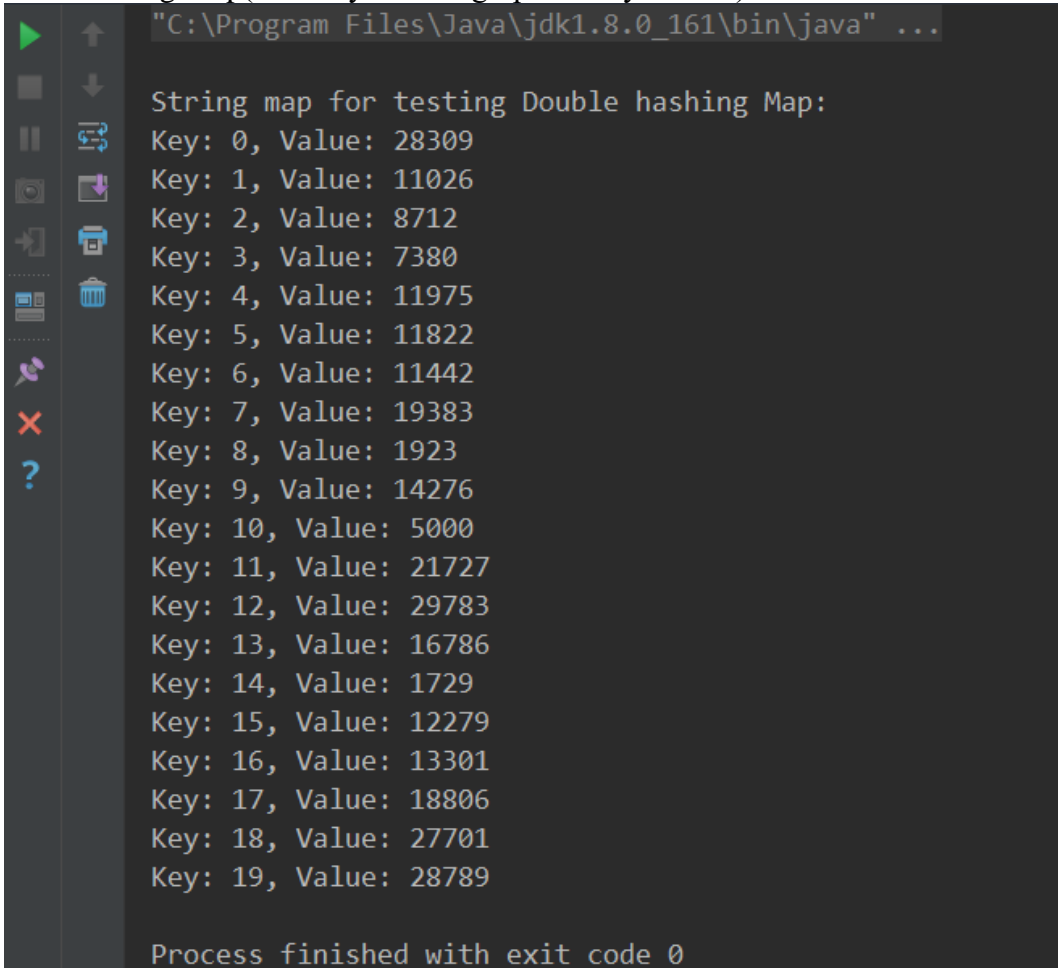
1.2 Test Cases

Test 1: Integer Map



```
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...  
Integer map for testing Double hashing Map:  
Key: 0, Value: 5712  
Key: 1, Value: 3054  
Key: 2, Value: 31721  
Key: 3, Value: 1913  
Key: 4, Value: 19295  
Key: 5, Value: 7428  
Key: 6, Value: 12996  
Key: 7, Value: 30271  
Key: 8, Value: 12733  
Key: 9, Value: 21635  
Key: 10, Value: 19479  
Key: 11, Value: 12216  
Key: 12, Value: 14507  
Key: 13, Value: 31129  
Key: 14, Value: 13730  
Key: 15, Value: 18444  
Key: 16, Value: 3332  
Key: 17, Value: 16019  
Key: 18, Value: 331  
Key: 19, Value: 29318  
  
Process finished with exit code 0
```

Test 2: String Map(Tüm sayılar string tipinde kaydedildi.)



```
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...  
  
String map for testing Double hashing Map:  
Key: 0, Value: 28309  
Key: 1, Value: 11026  
Key: 2, Value: 8712  
Key: 3, Value: 7380  
Key: 4, Value: 11975  
Key: 5, Value: 11822  
Key: 6, Value: 11442  
Key: 7, Value: 19383  
Key: 8, Value: 1923  
Key: 9, Value: 14276  
Key: 10, Value: 5000  
Key: 11, Value: 21727  
Key: 12, Value: 29783  
Key: 13, Value: 16786  
Key: 14, Value: 1729  
Key: 15, Value: 12279  
Key: 16, Value: 13301  
Key: 17, Value: 18806  
Key: 18, Value: 27701  
Key: 19, Value: 28789  
  
Process finished with exit code 0
```

2 Recursive Hashing Set

3 Sorting Algorithms

3.1 MergeSort with DoubleLinkedList

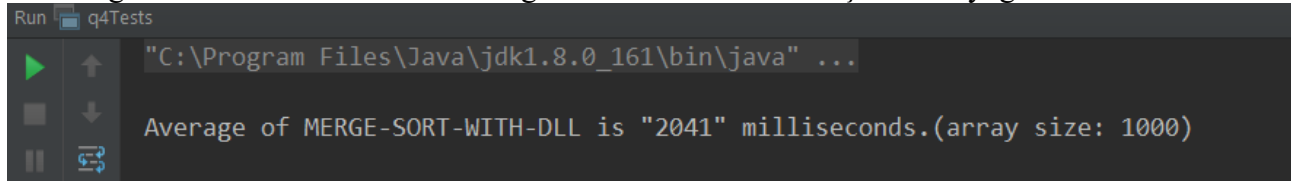
This part about Question3 in HW5

3.1.1 Pseudocode and Explanation

...

3.1.2 Average Run Time Analysis

10 tane 1000 elemanlık rastgele sayıların kayıtlı olduğu dizi oluşturuldu. Oluşturulan dizilerin her biri merge sort with double linked list algoritması kullanılarak küçükten büyüğe sıralandı.

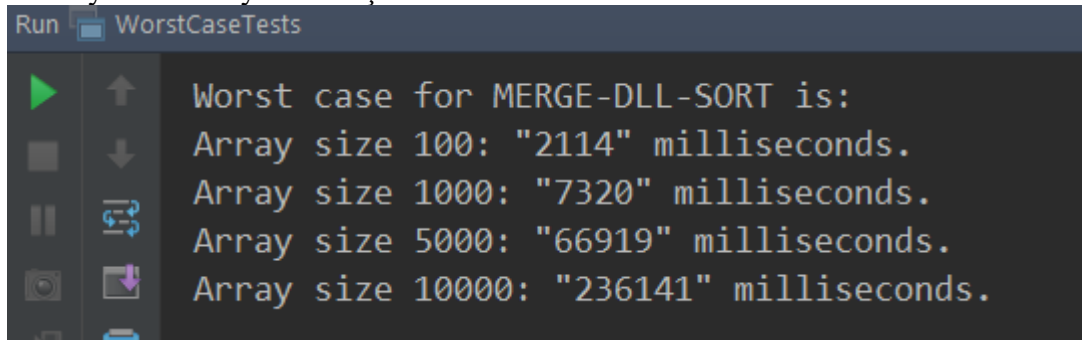


```
Run q4Tests
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...
Average of MERGE-SORT-WITH-DLL is "2041" milliseconds.(array size: 1000)
```

(Milisaniye cinsinden 10 farklı dizinin sıralanma süresinin ortalaması.)

3.1.3 Worst-case Performance Analysis

Merge sort için farklı büyüklüklerde diziler oluşturuldu. Diziler oluşturulurken ilk yarısı çift sayı, ikinci yarısı tek sayı olacak şekilde elemanlar eklendi.



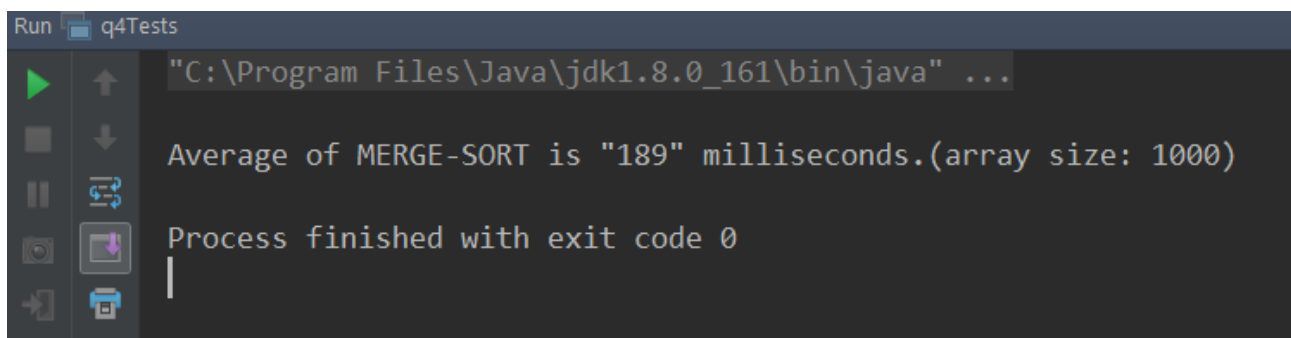
```
Run WorstCaseTests
Worst case for MERGE-DLL-SORT is:
Array size 100: "2114" milliseconds.
Array size 1000: "7320" milliseconds.
Array size 5000: "66919" milliseconds.
Array size 10000: "236141" milliseconds.
```

3.2 MergeSort

This part about code in course book.

3.2.1 Average Run Time Analysis

10 tane 1000 elemanlık rastgele sayıların kayıtlı olduğu dizi oluşturuldu. Oluşturulan dizilerin her biri merge sort algoritması kullanılarak küçükten büyüğe sıralandı.

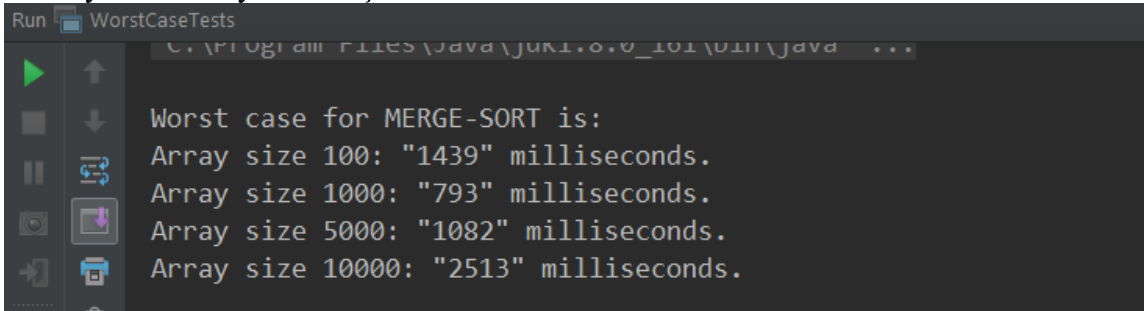


```
Run q4Tests
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...
Average of MERGE-SORT is "189" milliseconds.(array size: 1000)
Process finished with exit code 0
```

(Milisaniye cinsinden 10 farklı dizinin sıralanma süresinin ortalaması.)

3.2.2 Worst-case Performance Analysis

Merge sort için farklı büyüklüklerde diziler oluşturuldu. Diziler oluşturulurken ilk yarısı çift sayı, ikinci yarısı tek sayı olacak şekilde elemanlar eklendi.

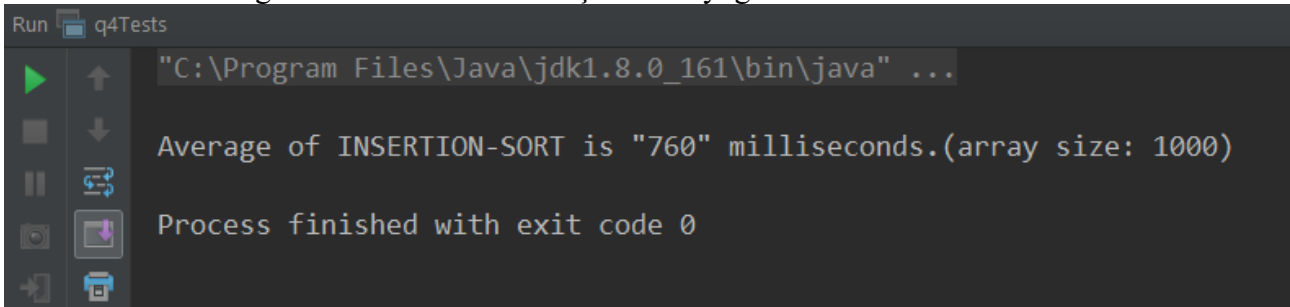


```
Run WorstCaseTests
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...
Worst case for MERGE-SORT is:
Array size 100: "1439" milliseconds.
Array size 1000: "793" milliseconds.
Array size 5000: "1082" milliseconds.
Array size 10000: "2513" milliseconds.
```

3.3 Insertion Sort

3.3.1 Average Run Time Analysis

10 tane 1000 elemanlık rastgele sayıların kayıtlı olduğu dizi oluşturuldu. Oluşturulan dizilerin her biri insertion sort algoritması kullanılarak küçükten büyüğe sıralandı.

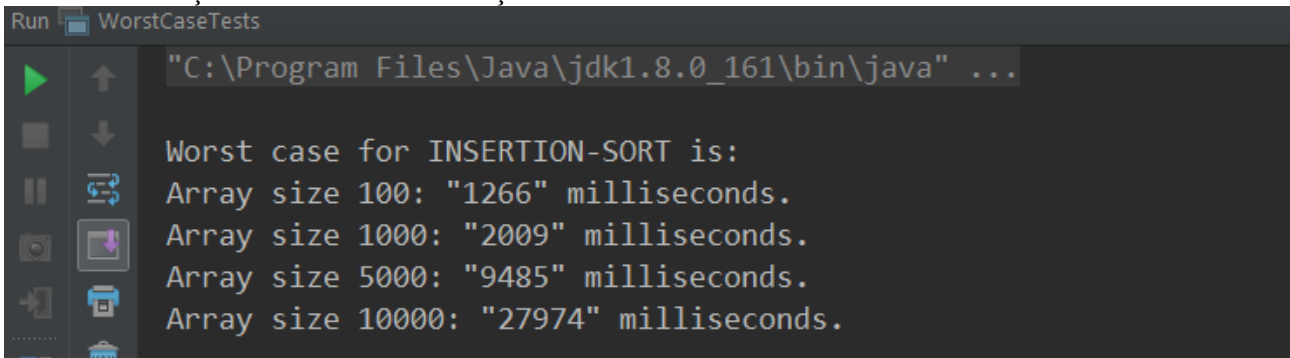


```
Run q4Tests
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...
Average of INSERTION-SORT is "760" milliseconds.(array size: 1000)
Process finished with exit code 0
```

(Milisaniye cinsinden 10 farklı dizinin sıralanma süresinin ortalaması.)

3.3.2 Worst-case Performance Analysis

Insertion sort için ters sıralı diziler oluşturuldu.

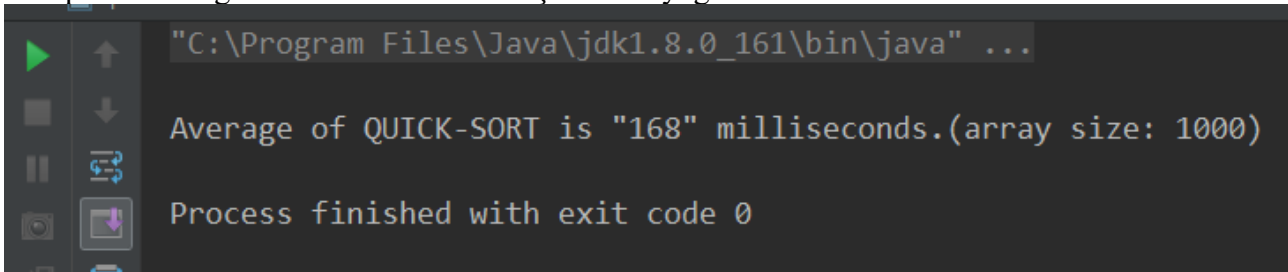


```
Run WorstCaseTests
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...
Worst case for INSERTION-SORT is:
Array size 100: "1266" milliseconds.
Array size 1000: "2009" milliseconds.
Array size 5000: "9485" milliseconds.
Array size 10000: "27974" milliseconds.
```

3.4 Quick Sort

3.4.1 Average Run Time Analysis

10 tane 1000 elemanlık rastgele sayıların kayıtlı olduğu dizi oluşturuldu. Oluşturulan dizilerin her biri quick sort algoritması kullanılarak küçükten büyüğe sıralandı.

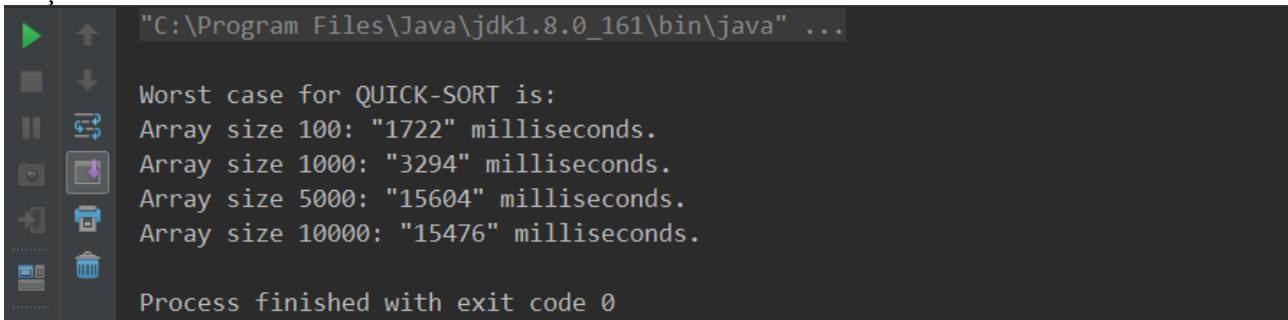


```
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...  
Average of QUICK-SORT is "168" milliseconds.(array size: 1000)  
Process finished with exit code 0
```

(Milisaniye cinsinden 10 farklı dizinin sıralanma süresinin ortalaması.)

3.4.2 Worst-case Performance Analysis

Yazdığım quick-sort algoritmasında ilk eleman pivot seçildiği için worst case için sıralı diziler oluşturdum.

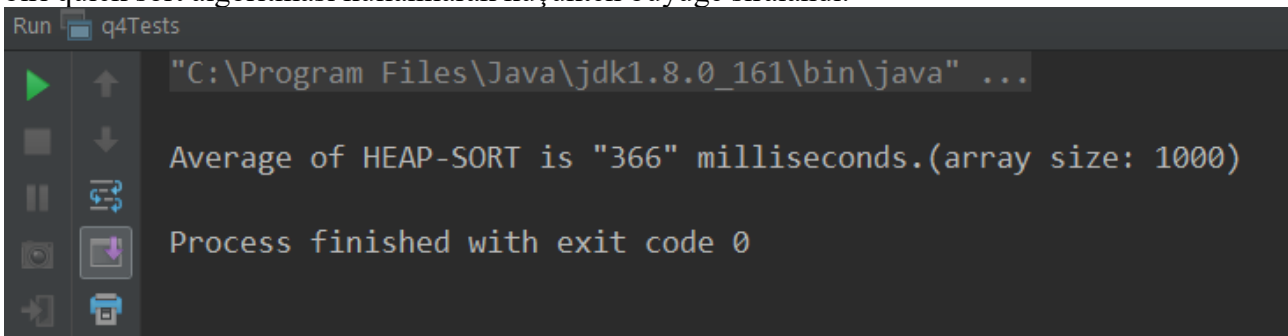


```
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...  
Worst case for QUICK-SORT is:  
Array size 100: "1722" milliseconds.  
Array size 1000: "3294" milliseconds.  
Array size 5000: "15604" milliseconds.  
Array size 10000: "15476" milliseconds.  
Process finished with exit code 0
```

3.5 Heap Sort

3.5.1 Average Run Time Analysis

10 tane 1000 elemanlık rastgele sayıların kayıtlı olduğu dizi oluşturuldu. Oluşturulan dizilerin her biri quick sort algoritması kullanılarak küçükten büyüğe sıralandı.



```
Run q4Tests  
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...  
Average of HEAP-SORT is "366" milliseconds.(array size: 1000)  
Process finished with exit code 0
```

(Milisaniye cinsinden 10 farklı dizinin sıralanma süresinin ortalaması.)

3.5.2 Worst-case Performance Analysis

Heap-sort'ta worst-case için ters sıralı listeleri test ettim.

```
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...  
  
Worst case for HEAP-SORT is:  
Array size 100: "1497" milliseconds.  
Array size 1000: "556" milliseconds.  
Array size 5000: "3237" milliseconds.  
Array size 10000: "1793" milliseconds.
```

4 Comparison the Analysis Results

