1. Requirement

Imagine you must handle information about *pattern calls*, which later should be displayed to the user. A pattern call is a tuple consisting of a unique integer identifier (“id”), a user defined name (“name”), a project relative path to the so called pattern file (“patternFile”) and a convenience flag, which states whether the pattern should be called or not (“called”). An example tuple is as follows:

42, “myPattern”, *src/patterns/Functional.pat*, false

Write a solution in [C++/Java] which holds pattern call tuples in memory, is able to store a set of tuples to a file (resp. read a set from a file), and answers the following queries:

- retrieve a pattern call with a specified identifier, e.g. 42
- list all pattern calls with a specified name, e.g. “myPattern”
- list all pattern calls with a specified path, e.g. “src/patterns/Functional.pat”
- list all pattern calls which **are** skipped, i. e. when the “called” flag is false
- list all pattern calls which **are not** skipped, i. e. when the “called ” flag is true

The number of tuples are not known beforehand, but are assumed to fit in-memory. In addition, the tuples will not be modified after initialization. The code you deliver should be **production ready**, i.e. maintainable and documented.

Finally, write a simple [C++/Java] program that demonstrates the implemented interfaces. It should create a few tuples with dummy data, store them in a file, read the tuples back from the file and check that the above mentioned query interfaces work correctly.

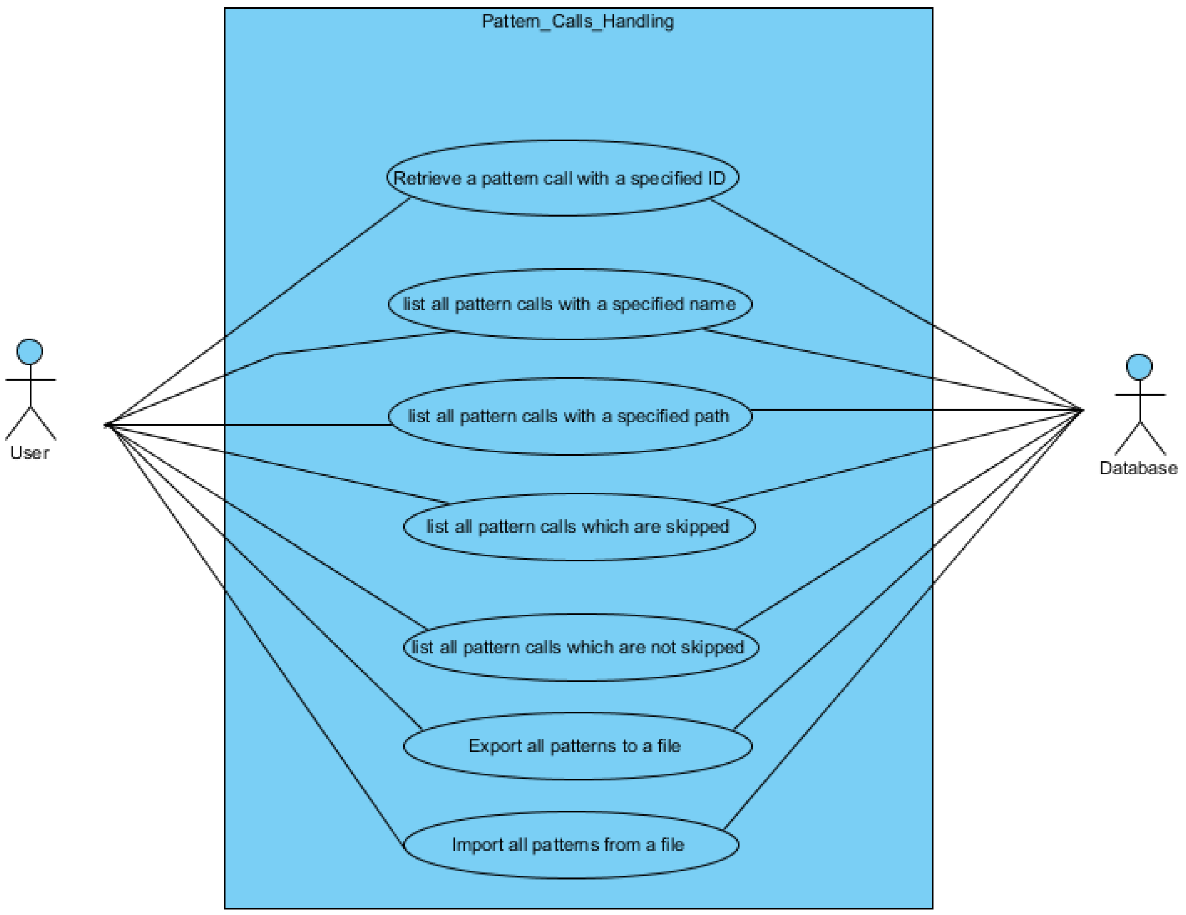
Questions to be answered:

- *What are the benefits of your design?*
- *Do you see improvement potential?*
- *What assumptions did you make and what trade-offs did you consider?*
- ***What is the complexity (Big-O notation) of the queries you provide?***
- *Which part of your solution took the most time (e.g. design, coding, documentation) and why?*

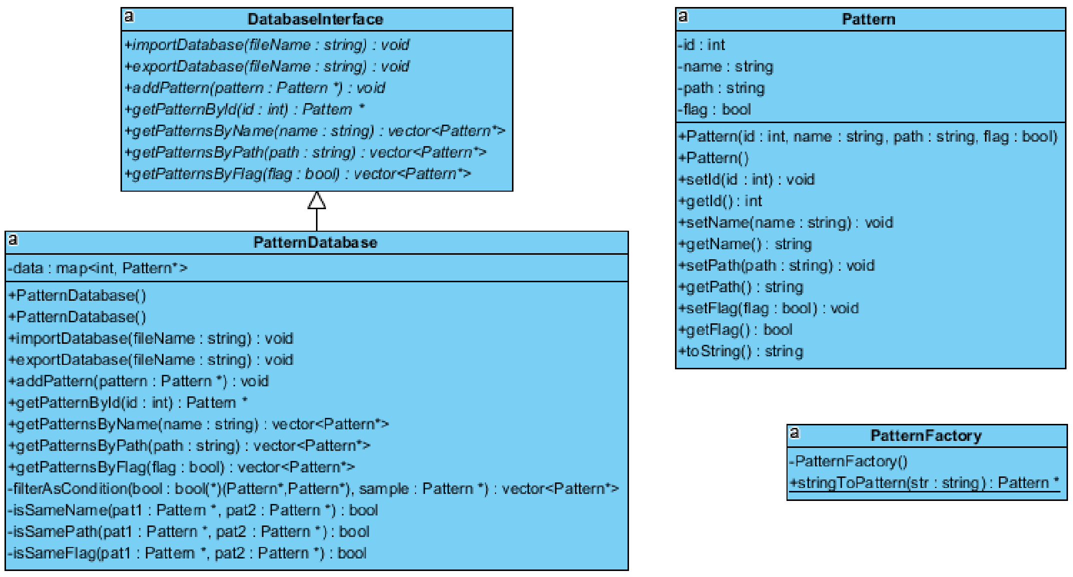
Please send us your complete solution in a zip file and the answers to the questions above. If you have any further questions please let us know. Otherwise, we look forward to seeing your solution!

2. Functional Design

User Case Diagram of program:

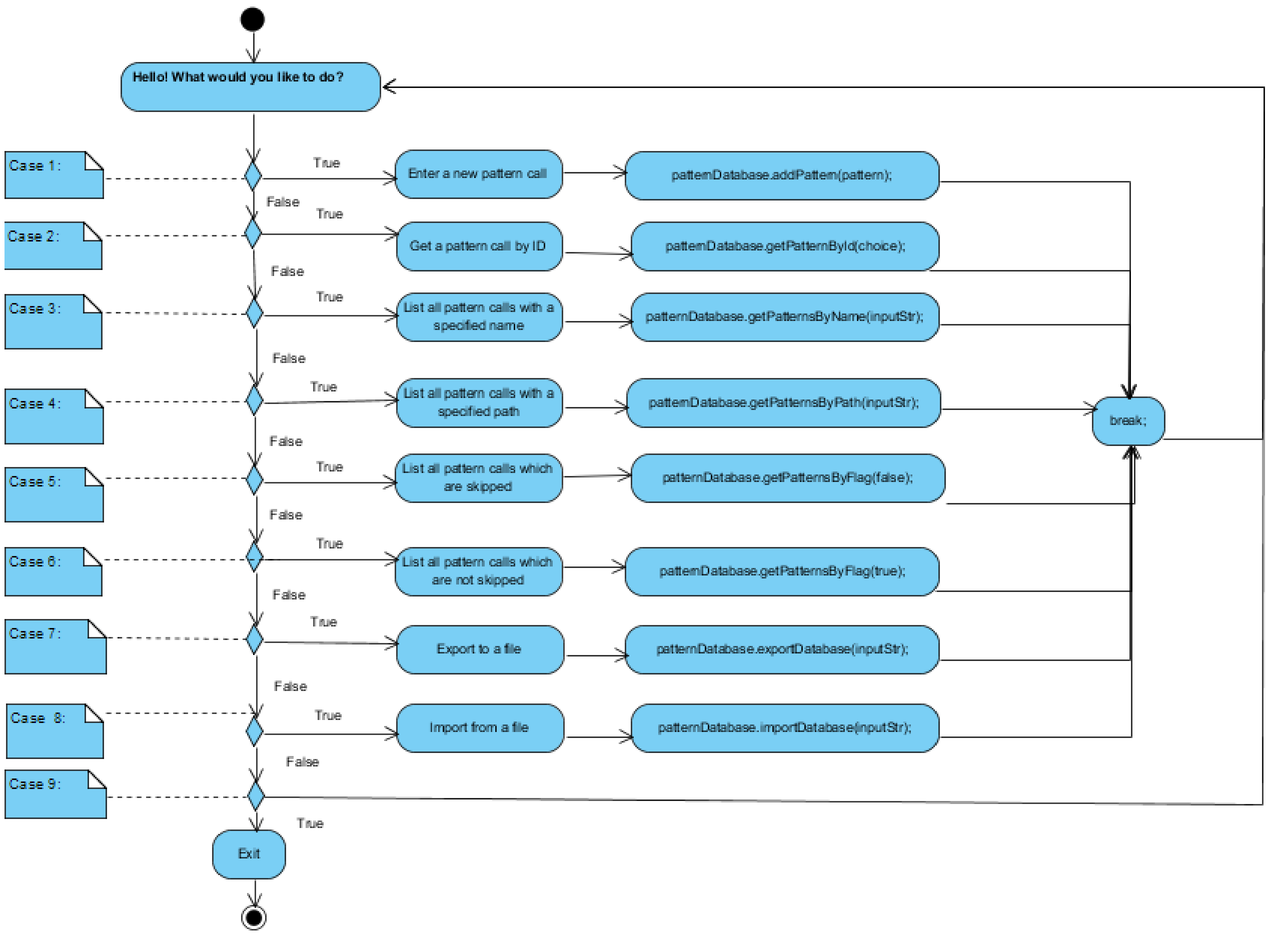

Pattern_Calls_Handling
Retrieve a pattern call with a specified ID
list all pattern calls with a specified name
list all pattern calls with a specified path
list all pattern calls which are skipped
list all pattern calls which are not skipped
Export all patterns to a file
Import all patterns from a file
User
Database

Class Diagram of program:


DatabaseInterface
+importDatabase(fileName : string) : void
+exportDatabase(fileName : string) : void
+addPattern(pattern : Pattern *) : void
+getPatternById(id : int) : Pattern *
+getPatternsByName(name : string) : vector<Pattern*>
+getPatternsByPath(path : string) : vector<Pattern*>
+getPatternsByFlag(flag : bool) : vector<Pattern*>
PatternDatabase
-data : map<int, Pattern*>
+PatternDatabase()
+PatternDatabase()
+importDatabase(fileName : string) : void
+exportDatabase(fileName : string) : void
+addPattern(pattern : Pattern *) : void
+getPatternById(id : int) : Pattern *
+getPatternsByName(name : string) : vector<Pattern*>
+getPatternsByPath(path : string) : vector<Pattern*>
+getPatternsByFlag(flag : bool) : vector<Pattern*>
-filterAsCondition(bool : bool(*)(Pattern*,Pattern*), sample : Pattern *) : vector<Pattern*>
-isSameName(pat1 : Pattern *, pat2 : Pattern *) : bool
-isSamePath(pat1 : Pattern *, pat2 : Pattern *) : bool
-isSameFlag(pat1 : Pattern *, pat2 : Pattern *) : bool
Pattern
-id : int
-name : string
-path : string
-flag : bool
+Pattern(id : int, name : string, path : string, flag : bool)
+Pattern()
+setId(id : int) : void
+getId() : int
+setName(name : string) : void
+getName() : string
+setPath(path : string) : void
+getPath() : string
+setFlag(flag : bool) : void
+getFlag() : bool
+toString() : string
PatternFactory
-PatternFactory()
+stringToPattern(str : string) : Pattern *

3. Implementation

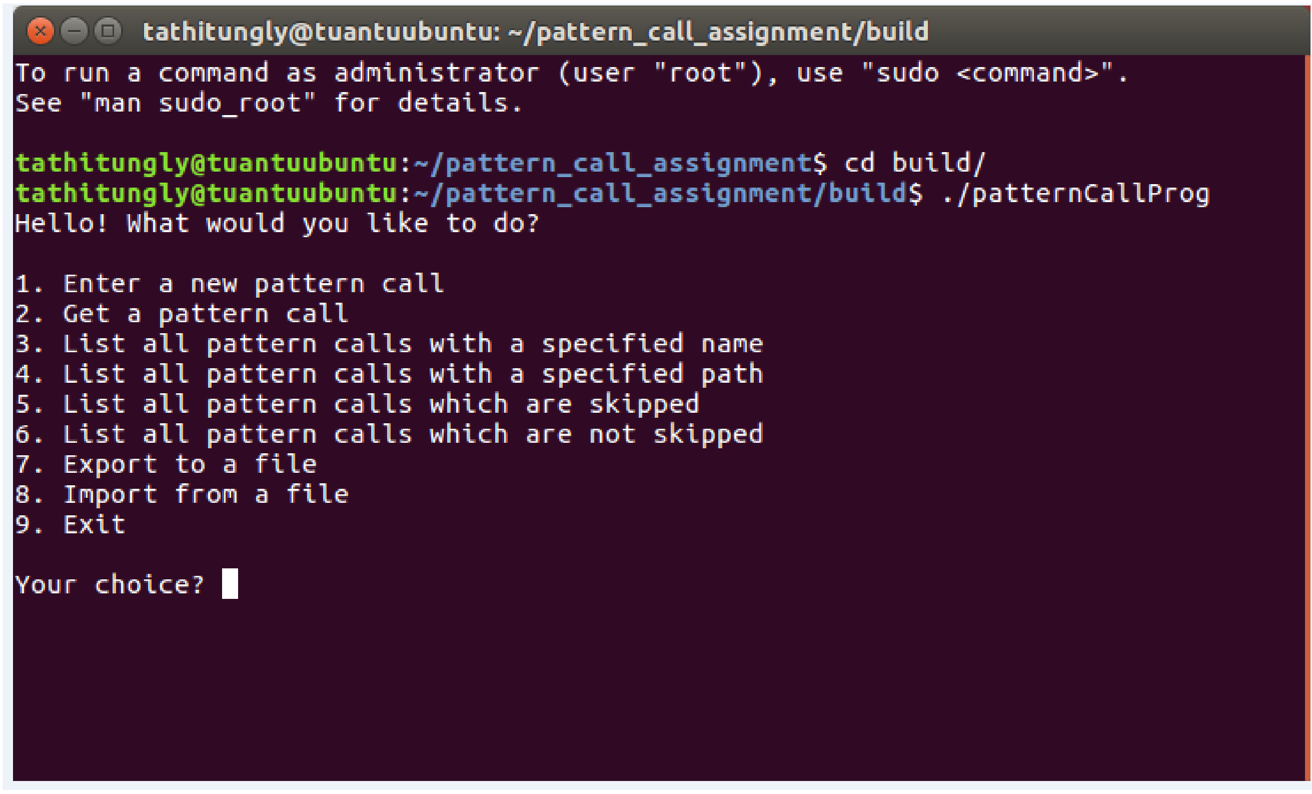
Compiler: G++ in Ubuntu16.04

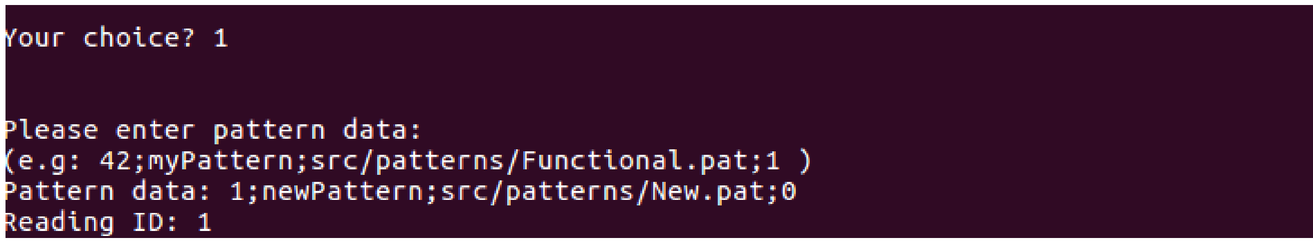


Hello! What would you like to do?
Case 1:
Case 2:
Case 3:
Case 4:
Case 5:
Case 6:
Case 7:
Case 8:
Case 9:
True
False
Enter a new pattern call
Get a pattern call by ID
List all pattern calls with a specified name
List all pattern calls with a specified path
List all pattern calls which are skipped
List all pattern calls which are not skipped
Export to a file
Import from a file
patternDatabase.addPattern(pattern);
patternDatabase.getPatternById(choice);
patternDatabase.getPatternsByName(inputStr);
patternDatabase.getPatternsByPath(inputStr);
patternDatabase.getPatternsByFlag(false);
patternDatabase.getPatternsByFlag(true);
patternDatabase.exportDatabase(inputStr);
patternDatabase.importDatabase(inputStr);
break;
Exit

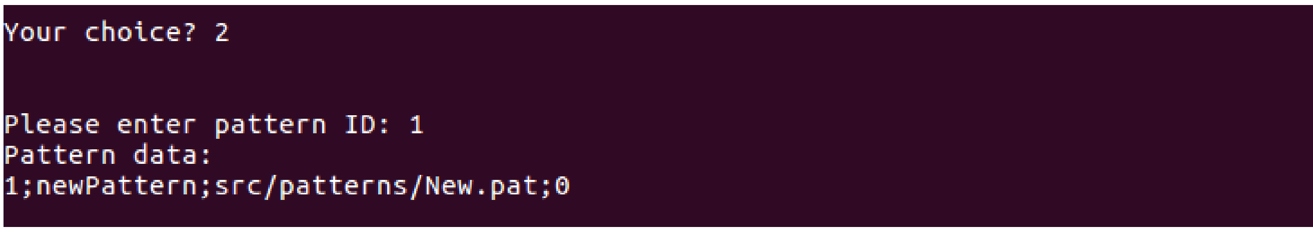
4. Result

https://github.com/tathitungly/pattern_call_assignment


```
tathitungly@tuantuubuntu: ~/pattern_call_assignment/build
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tathitungly@tuantuubuntu:~/pattern_call_assignment$ cd build/
tathitungly@tuantuubuntu:~/pattern_call_assignment/build$ ./patternCallProg
Hello! What would you like to do?

1. Enter a new pattern call
2. Get a pattern call
3. List all pattern calls with a specified name
4. List all pattern calls with a specified path
5. List all pattern calls which are skipped
6. List all pattern calls which are not skipped
7. Export to a file
8. Import from a file
9. Exit

Your choice? 
```

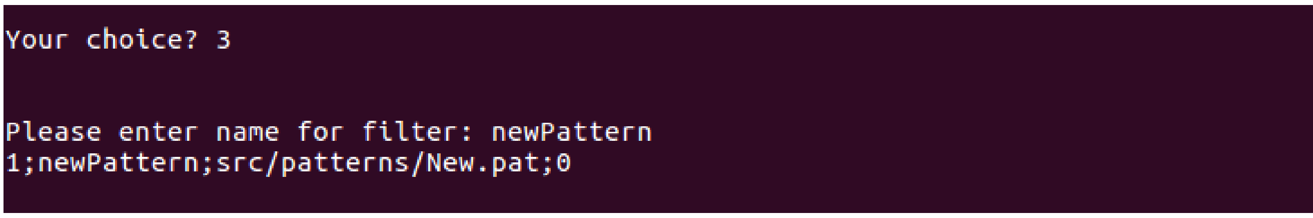


```
Your choice? 1


Please enter pattern data:
(e.g: 42;myPattern;src/patterns/Functional.pat;1 )
Pattern data: 1;newPattern;src/patterns/New.pat;0
Reading ID: 1
```

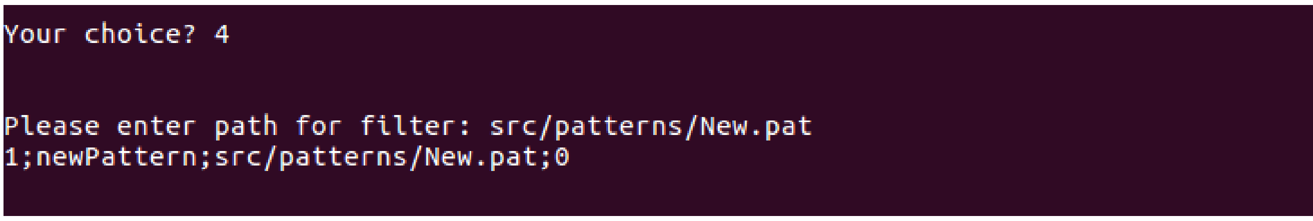


```
Your choice? 2


Please enter pattern ID: 1
Pattern data:
1;newPattern;src/patterns/New.pat;0
```

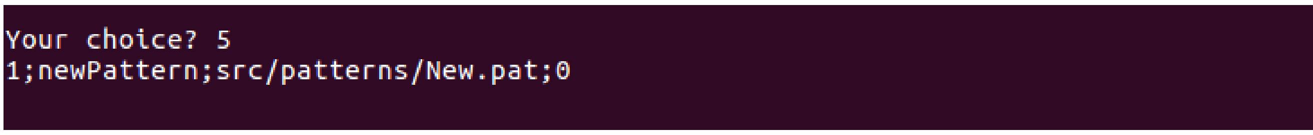
as



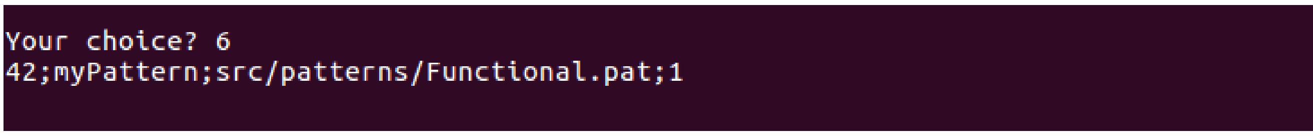
```
Your choice? 3


Please enter name for filter: newPattern
1;newPattern;src/patterns/New.pat;0
```



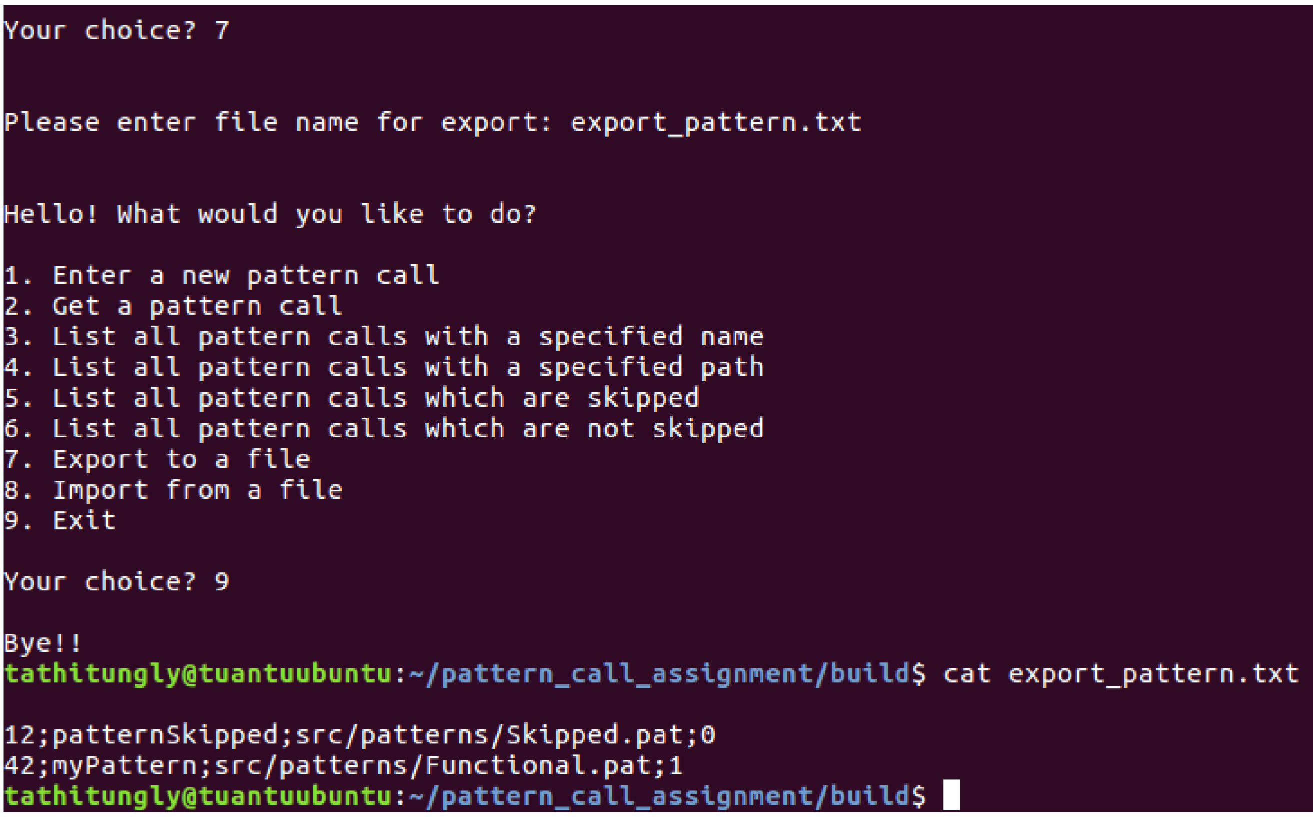
```
Your choice? 4


Please enter path for filter: src/patterns/New.pat
1;newPattern;src/patterns/New.pat;0
```



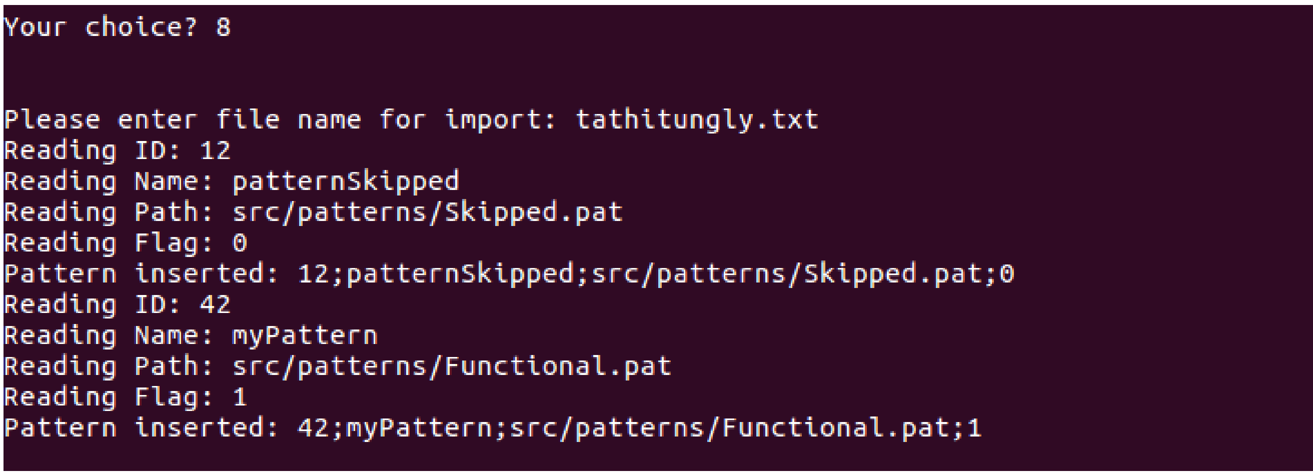
```
Your choice? 5
1;newPattern;src/patterns/New.pat;0
```



```
Your choice? 6
42;myPattern;src/patterns/Functional.pat;1
```



```
Your choice? 7


Please enter file name for export: export_pattern.txt


Hello! What would you like to do?

1. Enter a new pattern call
2. Get a pattern call
3. List all pattern calls with a specified name
4. List all pattern calls with a specified path
5. List all pattern calls which are skipped
6. List all pattern calls which are not skipped
7. Export to a file
8. Import from a file
9. Exit

Your choice? 9

Bye!!
tathitungly@tuantuubuntu:~/pattern_call_assignment/build$ cat export_pattern.txt

12;patternSkipped;src/patterns/Skipped.pat;0
42;myPattern;src/patterns/Functional.pat;1
tathitungly@tuantuubuntu:~/pattern_call_assignment/build$
```



```
Your choice? 8


Please enter file name for import: tathitungly.txt
Reading ID: 12
Reading Name: patternSkipped
Reading Path: src/patterns/Skipped.pat
Reading Flag: 0
Pattern inserted: 12;patternSkipped;src/patterns/Skipped.pat;0
Reading ID: 42
Reading Name: myPattern
Reading Path: src/patterns/Functional.pat
Reading Flag: 1
Pattern inserted: 42;myPattern;src/patterns/Functional.pat;1
```

5. Discussion

- ***What are the benefits of your design?***

  The benefits of my design are:

  1. It can cover the requirement of retrieving data by pattern calls
  2. The class “DatabaseInterface” is an abstract class, if we want to change to completely new database format, we can use “DatabaseInterface”, if we want to override only one method (ex: importDatabase()), we only do it in “PatternDatabase”
  3. Currently, STD map is used for storing pattern data, the ID duplication is not required to check.
  4. Using standard C++ libraries, easy to build and run in several platforms

- ***Do you see improvement potential?***

  Yes, there are some rooms to improvement in my solution”

  1. The current implementation only support one format of pattern which using “;” to separate different data fields. The function toString() in class “Pattern” and stringToPattern() in class “PatternFactory” should be moved to the abstract class “PatternDatabase”, so that the pattern format would be more flexible (with blank space, comma, …)

2. Add history of current queries to a map for speeding up when users want to query the similar pattern in future.

- ***What assumptions did you make and what trade-offs did you consider?***
  Using the program with assumptions:
  1. Users muss input corrected format of pattern
  2. The flag of pattern should be 0 (False) or 1 True

  I have considered the trade-offs between speed of the queries and memory allocation.
  Example:
  Currently using one Map to store all patterns, key is ID
  Data = {ID1: Pattern 1, ID2 : Pattern 2, ID3: Pattern 3,…};
  [ID1] = Pattern 1 with Flag = 0
  [ID2 = Pattern 2 with Flag = 1
  [ID3] = Pattern 3 with Flag = 0
  …
  To speed up the query, one more Map could be used to store the patterns with Flag = 0 and Flag = 1
  MapFlag[0] = [Pattern1, Pattern 3];
  MapFlag[1] = [Pattern2];

- ***What is the complexity (Big-O notation) of the queries you provide?***
  The complexity of the queries is as following:
  Retrieve a pattern call with a specified ID: O (logN)
  List all pattern calls with a specified name: O (N)
  List all pattern calls with a specified path: O (N)
  List all pattern calls which are skipped: O (N)
  List all pattern calls which are not skipped: O (N)

- ***Which part of your solution took the most time (e.g. design, coding, documentation) and why?***
  During this assignment, the part of coding took me most time to do because I have one year (from Jan 2016 to Feb 2017) not working in C++ project. It took time to set up environment , review the usage of C++ libraries…