**Background:** Changes are inevitable during the evolution of a software system. Such changes can be triggered due to fixing bugs or adding new features. If a software system depends on reusable components (such as Application Programming Interfaces (APIs) of libraries and frameworks), changes to those components **also lead to the changes in software to use the new version of the components. This requires improving the design of existing code through a set of behavior-preserving changes, also known as refactoring**. Refactoring detection is an active research area, and more detail of refactoring detection techniques can be found in the following papers. In this assignment, you will focus on detecting source code changes that lays the foundation of refactoring detection.

More information on refactoring:

1. https://martinfowler.com/books/refactoring.html

Reference Papers:

1. Automated Detection of Refactoring in Evolving Components
2. Automated detection of api refactorings in libraries
3. Identifying Refactorings from Source-Code Changes

**Task:** Given two different versions of a source file, determine the mapping between methods in those two files. The input to your program should be two different versions of a file written in either python or java language (you can select one of the two languages). For simplicity consider that each file contains only one class consisting of a set of methods.

Methods can be added or deleted in the new version, method in the old version can be renamed or modified (e.g., lines added, deleted, parameters or return type of the methods can also be changed) in the new version.

**Hints:** If you can identify the method bodies along with their signatures, you can apply similarity measures to determine which method in the old versions is similar to which method in the new version. This can help you to determine the mapping. The reference papers will help you to understand the process better. You can reuse any exiting technique or use your own idea or heuristics.

Sample input:

|  |  |
| --- | --- |
| File Name: v1. java | File Name: v2.java |
| public class A{  public void doRevertToSaved(){  //code goes here  ….  performRevertOperation (100);  }  public void performRevertOperation (int timeLimit){  //code goes here  }  public static void main (string args[]){  //code goes here  }  protected void free(){  //code goes here  }  } | public class A{  public void doRevertToSaved(){  //code goes here  … performRevert(50);  }  public void performRevert (){  //code goes here  }  public static void main (string args[]){  //code goes here  }  void init(){  //code goes here  }  } |

Sample output:

v1.A. doRevertToSaved() mapped to v2.A. doRevertToSaved()

v1.A. performRevertOperation (int timeLimit) mapped to v2.A. performRevert ()

v1.A. main (string args[]) mapped to v2.A. main (string args[])

v1.A.free() is deleted in the new version (Note: because it is in the old version and you do not find any mapping here)

v2.A.init() is added in the new version (Note: because it is in the new version and you do not find any mapping here)

like Java AST parser and ADT parser