Garbage Collectors

Objective

In this lab, you will implement two of the known algorithms used in Garbage Collectors:

- Mark & Compact GC
- Copy GC

The input to your program will be

- File **heap.csv** : this is a comma separated file with three columns. Each line represents the information about a single allocated object. This object may be used or not used.
  - *object-identifier:* a unique 6 digits identifier of the allocated objects.
  - *memory-start:* the index of the first byte in heap memory representing this object
  - *memory-end:* the index of the last byte in heap memory representing this object
- File **roots.txt**: this is a text file that lists object-identifiers that are currently in use. Any object that can not be reached directly or indirectly from objects listed in this file should be considered as a garbage. Each line in this file contains a single object-identifier.
- File **pointers.csv**: this file stores the dependencies between different objects. It is a comma separated file with two columns
  - *parent-identifier:* a unique identifier for the parent object
  - *child-identifier:* a unique identifier for the child object referenced by the parent

The output of your program will be

- File **new-heap.csv**: this is a comma separated file with the same structure of the **heap.csv** showing the new memory layout after running the garbage collector

As you will need to traverse linked objects represented by the **pointers.csv** file, the child traversal must process the children objects in the order it appears in the pointers.csv file. For example, if we have the following contents in the pointers.csv file

```
111111,222222
111111,333333
111111,444444
222222,555555
555555,666666
```

Then your program must process 222222 before 333333. This assumption is just needed to have the same output for all the students.

Deliverable

- Source code of the implementation of the two algorithms
- Two executables Jars each of them represents one of the algorithms. The jar accepts four arguments:
    - The first three arguments are the absolute paths of heap.csv, roots.txt, and pointers.csv.
    - The last argument is the absolute path in which the new-heap.csv file will be saved to
- 4 sets of files represents 8 runs of your program. Each set of the files will contain 5 files: the three input files, and two heap layouts using the two algorithms.

Delivery

- Submission of the assignment using Microsoft teams course page
- Group of 2 students
- Due Date is 4 June