1 Introduction

1.1 Assignment Rules

1. This assignment is meant to be solved individually and not in groups. That means the assignment is evaluated/graded individually.
2. When you believe your assignment is ready to be evaluated/graded, upload your solution using Moodle's submission system, and then it will get assessed after some days by us. Please refer to 2.5 for the submission structure.

2 Description

2.1 Task objective

Your objective is to simulate contiguous memory allocation (see Operating Systems Concepts, mainly chapter 9.2.2.) using all the first-fit, best-fit, and worst-fit strategies.

The result shall be a memory layout consisting of allocated blocks and holes (free blocks). Also, external fragmentation shall be computed according to the following formula:

$$fragmentation = 1 - \frac{Largest\ block\ of\ free\ memory}{Total\ free\ memory}$$

Also, a brief report describing your implementation strategies shall be submitted (i.e., your thoughts when constructing the application).

2.2 Task implementation

To control the order in which allocation, deallocation, and compaction of blocks occurs, an input file shall be used to list actions (see 2.3.1) to be performed. This means that the file is to be read by your program running the actions line by line. Simulation results shall be stored in one or more files according to 2.3.2.

If there is not enough memory to allocate a block, the program should skip the instruction that would lead to an error and continue the execution with the following statement, if any. The same applies if a deallocation instruction refers to a non-existing ID. Note that all the occurred errors must be documented and printed in the output file(s).

Also, all file paths shall be relative to your working directory!

See 2.4 for an execution example (input to output).

2.3 File formats

2.3.1 Input

There are four commands:

Command	Parameter 1	Parameter 2	Description
A	Block id	Amount of memory (bytes)	Allocate
D	Block id	Not used	Deallocate <block id>
C	Not used	Not used	Compact memory, meaning move all allocated block towards the lowest address.
O	Not used	Not used	Output the current status of the memory in an intermediate output

			file. See 2.3.2 for detailed information.

Rules:

1. If a command has parameters, they are separated by a ';' character.
2. There can be only one command per line.
3. First line in file is always an integer describing max memory size.
4. Any line thereafter is a command.

File content example:

```
1000        // Max number of bytes in memory
A;0;100     // Allocate 100 bytes to a block identified by 0
A;1;100
A;2;500
D;1         // Deallocate memory for block 1
A;3;200
D;2
O           // Produce an intermediate output file
A;4;1500
D;4
D;5
```

2.3.2 Output

An output file contains the results from a simulation or an intermediate representation of the memory during it. Its structure (layout) is described below.

A final output file name uses the input file name with the extension <.out>. An intermediate (e.g., generated after invoking the O command) output file name uses the input file name with the extension <.out#>, where # is the placeholder for an incremental integer. So, if the input file is 'scenario1.in', then the final output file is 'scenario1.out', while all the intermediate outputs will be named 'scenario1.out1', 'scenario1.out2', and so on.

Content structure

Note 1: Anything within a <>-pair below shall be replaced by actual values. ':' means additional lines. E.g., <block id>;<start address>; <end address> is replaced with, for instance, 0;200;300.

Note 2: Allocated blocks are listed by ascending block id.

Note 3: Free blocks are listed by ascending start address.

Note 4: In case of an allocation error when executing an A command, the following format shall be used under the Errors section:

```
<type of instruction>;<instruction number>;<max available blocks>
```

Where instruction number refers to the number of the command that triggered the error (that is equal to its line number – 1) and the max available blocks refer to the maximum amount of allocation space (NOT the total free space!) available when the error occurred. The type of instruction must be 'A'.

In case of a deallocation error when executing a D command, the following format shall be used under the Errors section:

```
<type of instruction>;<instruction number>;<reason for failure>
```

Where instruction number refers to the number of the command that triggered the error (that is equal to its line number – 1) and the reason for failure should be set to 0 if the ID to deallocate was never attempted to be allocated or to 1 if the ID to deallocate was previously tried to be allocated but resulted in an error. The type of instruction must be 'D'.

If the execution did not return any errors (or the intermediate output has no errors until that point), just output 'None' under the Errors section.

Note 5: Fragmentation is ALWAYS a 6 decimals number. This means that if your value is for example 0.5, it should be printed in the output file as 0.500000. Standard rounding rules apply if your value has more than 6 digits for decimals.

Structure:

```
First fit
Allocated blocks
<block id>;<start address>; <end address>
:
Free blocks
<start address>;<end address>
:
Fragmentation
<fragmentation rounded to 6 decimals>
Errors
<type of instruction>;<instruction number>;<3rd parameter>

:

Best fit
Allocated blocks
<block id>;<start address>; <end address>
:
Free blocks
<start address>;<end address>
:
Fragmentation
<fragmentation rounded to 6 decimals>
Errors
<type of instruction>;<instruction number>;<3rd parameter>

:


Worst fit
Allocated blocks
<block id>;<start address>; <end address>
:
Free blocks
<start address>;<end address>
:
Fragmentation
<fragmentation rounded to 6 decimals>
Errors
<type of instruction>;<instruction number>;<3rd parameter>

:
```

2.4 Execution example

Scenario1.in:

```
1000
A;0;100
A;1;100
A;2;500
D;1
A;3;200
D;2
O
A;4;1500
D;4
D;5
```

Scenario1.out1 (first-fit shown only): // this is generated after the O command at line 8

```
First fit
Allocated blocks
0;0;99
3;700;899
Free blocks
100;699
900;999
Fragmentation
0.142857
Errors
None

Best fit
:
:
```

Scenario1.out (first-fit shown only): // this is the final output, produced after every command is processed

```
First fit
Allocated blocks
0;0;99
3;700;899
Free blocks
100;699
900;999
Fragmentation
0.142857
Errors
A;8;600
D;9;1
D;10;0

Best fit
:
:
```

2.5 Submission structure

Your submission MUST be composed of a report in .pdf format and a .zip file containing your executable source code along with the instructions for building it. Attaching your output files in the submission is unnecessary, as your code will be executed for every desired input file and the produced output files will be evaluated. Submissions that fail to comply with this format (e.g., report in .docx format, no building instructions, report placed inside the .zip file, …) will see a slight reduction in their overall score.