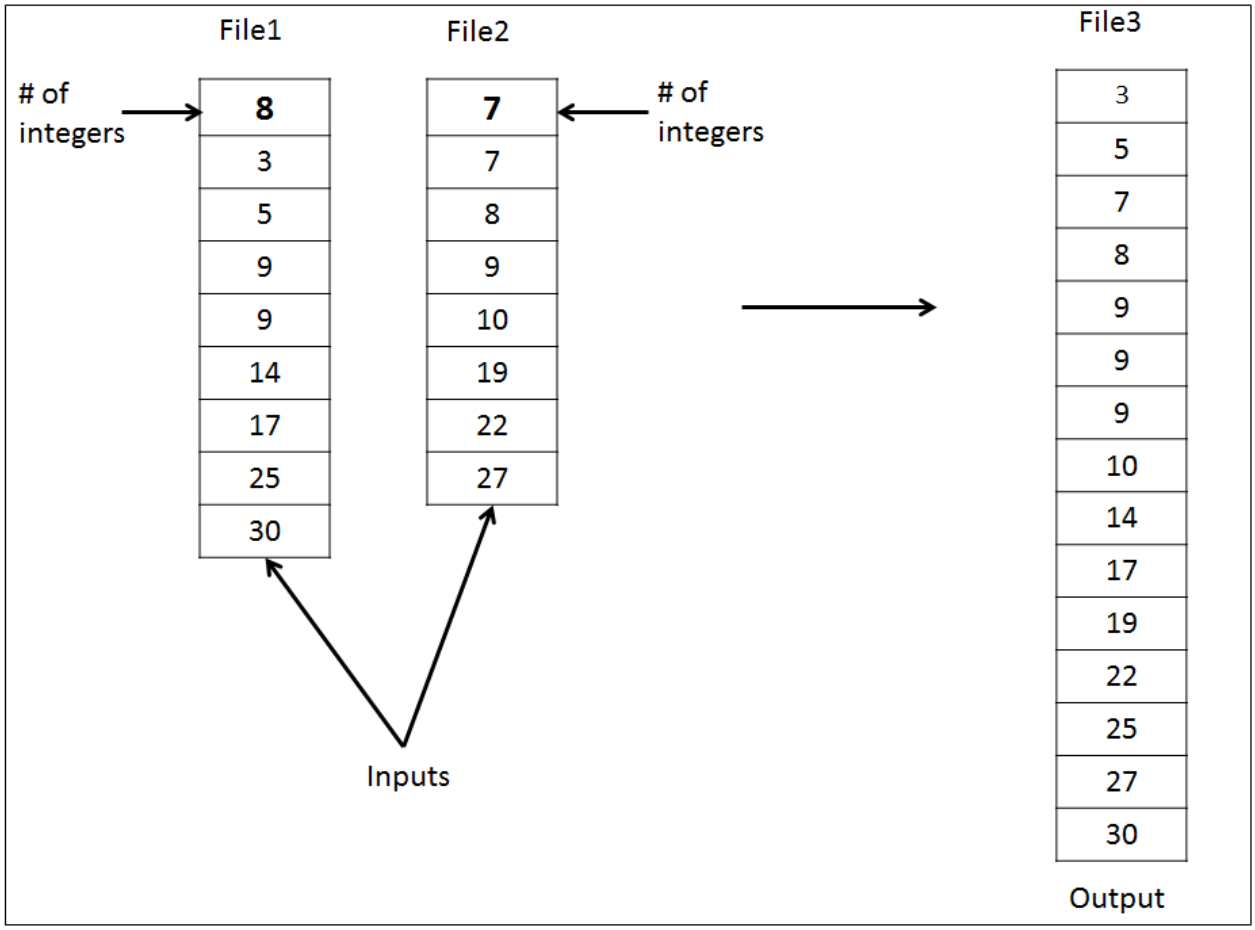
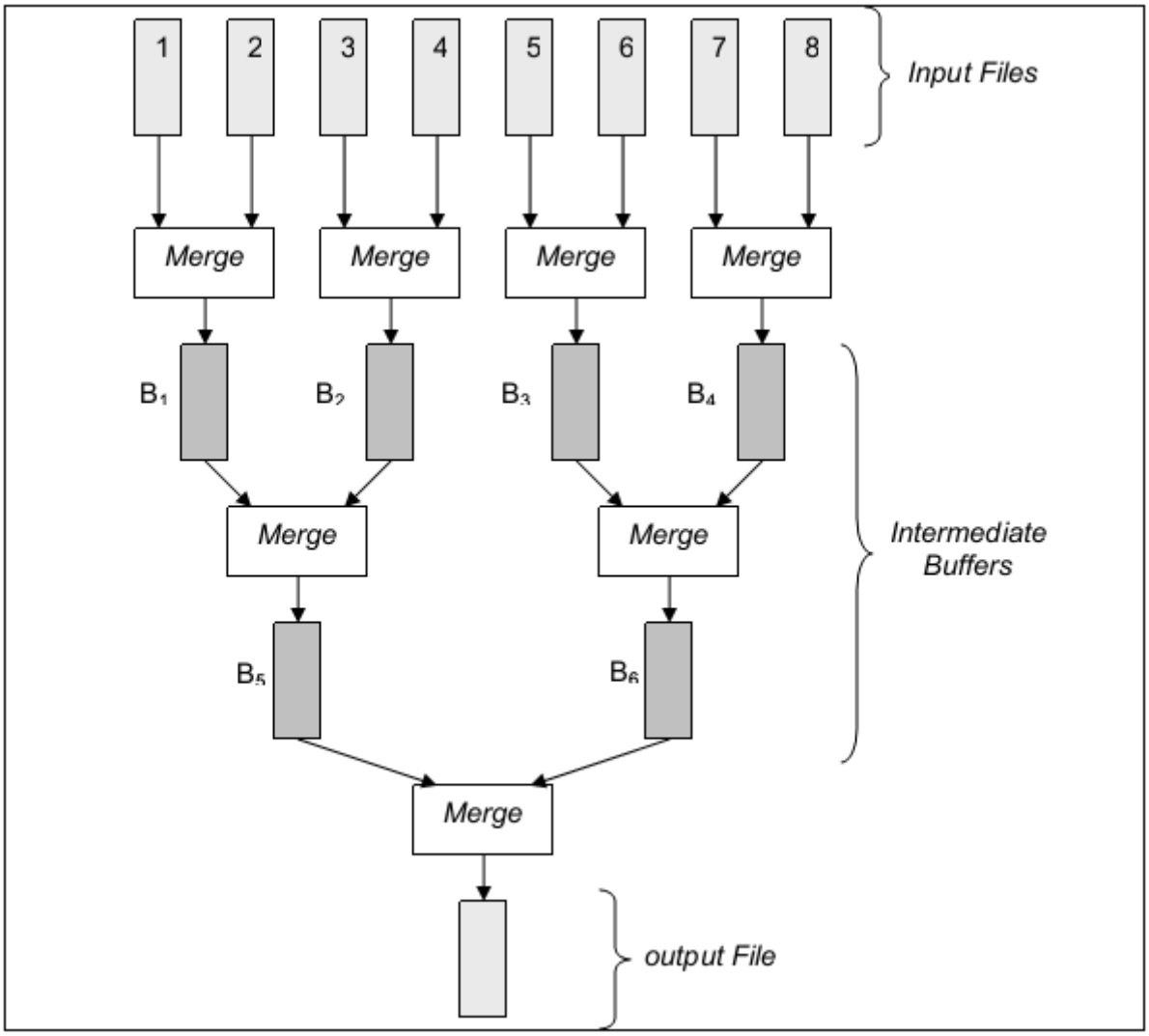
# CSE 4034 – ADVANCED UNIX PROGRAMMING
## Project # 2
## DUE DATE: 18/06/2021 - 23:59

Your task is to implement a parallel merge of sorted integers stored in N input files. The input files will contain sorted integers in ascending order. Each line of an input file will contain exactly one integer. The first line of each input file will contain total number of integers in that file. For example merging two input files into one output file will be like the following:




The number of input files can be 2, 4, or 8. In more than 2 files, you will merge the files in several phases. A merge will be done always between two files (no more than that). The idea is shown in the figure below. There are eight input files, each of which stores sorted integers. Then these files will be merged into four intermediate buffers. The content of these four buffers will merged into two other intermediate buffers. The content of these two other buffers will be merged into a final

output file. Note that the buffers B1 through B6 are shared buffers. While one merge operation is using a buffer for output, another merge operation is using that buffer for input.




You will use concurrent threads to implement the merge operations. For example in the figure above, there would be seven threads that are created by a main thread. These seven threads should be created at the beginning based on the total number of files and each should perform one merge operation as shown in the figure concurrently. A thread, while doing a merge operation, will take the input from either two files or two shared buffers. A thread will write the output to either a final output file or to another shared buffer. You are required to create all of your threads at the beginning.

In case of 2 input files, only one thread is adequte. In case of 4 files, you should create 3 threads in total.

The shared buffers will be implemented as global structures in the main process, so that all threads can share and access them. Since multiple threads will be accessing and modifying a shared buffer, you should use *suitable synchronization primitives* to provide *mutual exclusion* and *synchronization*. Otherwise, your results may be inconsistent and wrong.

The invocation of your program will be as follows:

```
$ gcc project2.c –o project2.out -lpthread
$ ./project2.out  –n 4 file1 file2 file3 file4 -o outputfile
```

The *project2.out* is the name of your program. The *–n* option specifies the number of input files. Then comes the name of the input files. The name of the output file is specified using the *–o* option.

**Bonus:** You will get bonus points if you present a comparison analysis for single-threaded implementation with multithreaded implementation. For the example above, you can use just a single thread to perform all merge operations. Then, you should compare the performance of the single-threaded run with the multithreaded run. You can use **`clock_gettime()`** to measure the performance of your program. Please comment on the results in your report.

**Notes:**

- The project will be done in Linux operating system using C programming language.
- You must use PThread library and synchronization appropriately in your code.
- Take into account materials and examples covered in the lectures.
- You can work in groups of 2 people.
- Please write a minimum 2-pages long project report that describes how you implemented the project and add several screenshots that show sample runs.
- We will use tools that automatically detect plagiarism among the submissions!

- In case of any form of copying and cheating on solutions, you will get **FF** grade from the course! You should submit your own work. In case of any forms of cheating or copying, both giver and receiver are equally culpable and suffer equal penalties.
- Please zip and submit your files using filename Student1Number_Student2Number_Project1.zip (ex: 150713852_150713098_Project1.zip) to Canvas system (under Assignments tab).
- No late submission will be accepted.