Final

Note

This is an open-book exam. You may consult any book, paper, note, or on-line resource, but discussions with others (in person or via a network) are strictly forbidden.

How to Submit Your Answers

Please use a word processor or scan hand-written answers to produce a single PDF file. Name your file according to this pattern: "r107250xx-final". Upload the PDF file by the due time to the NTU COOL course site for Software Development Methods 2022.

Problems

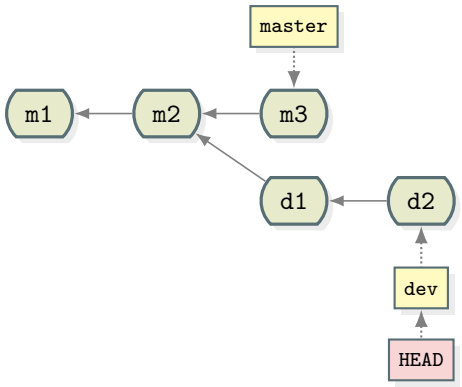
1. Answer the following questions.

   (a) (3 %) What git command is used to check the state of the working directory and the staging area?

   (b) (3 %) Assume `main.c` has been changed since it was checked out from a git repository. After `git add main.c`, what state does `main.c` reside in?

   (c) (3 %) Follow 1b. After `git commit`, what state does `main.c` reside in?

   (d) (8 %) Consider a git history with two branches, namely master and dev (pointed to by HEAD). Write down at most two git commands to merge dev into master, and also draw the resulting git history.



   (e) (3 %) Follow 1d. Is the merge fast-forward?

2. Suppose you are working on an audio processor class. The class has the following method that compresses audio files in parallel using multiple threads:

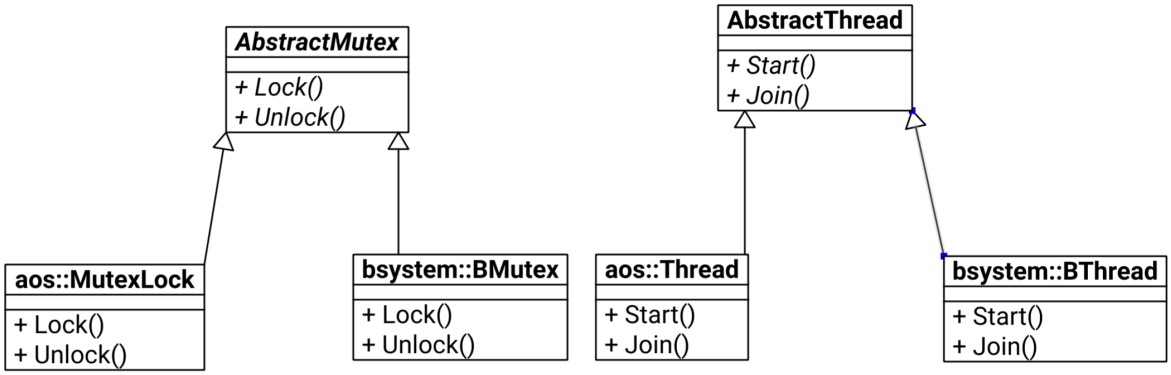
```
using namespace std;
void AudioProcessor::ParallelCompress(
    const vector<AudioFile>& files,
    int nThreads) {
  vector<unique_ptr<aos::Thread>> threads(
      nThreads);
  // Creates a aos::MutexLock instance.
  unique_ptr<aos::MutexLock> lock(
      new aos::MutexLock());

  int fIndex = files.size();
  for (int i = 0; i < nThreads; ++i) {
    // Create a aos::Thread instance.
    threads[i].reset(new aos::Thread());
    // Run the lambda on a thread.
    threads[i]->Start([&](){
      while (true) {
        // Get an unprocessed AudioFile instance.
        lock->Lock();
        if (fIndex == files.size()) {
          // Returning from the lambda will
          // terminate the thread.
          return;
        }
        const auto& file = files[fIndex++];
        lock->Unlock();

        // Compress the AudioFile instance.
        DoCompress(file);
      }
    });
  }

  // Wait until all threads are terminated.
  for (auto& thread : threads) {
    thread->Join();
  }
}
```

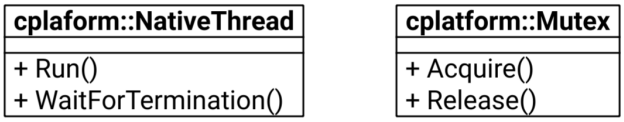
Now you are supporting a new platform named bsystem that provides the bsystem::BThread and bsystem::BMutex classes for manipulating threads and mutexes. Here is the class diagram of the threading classes:



(a) (3 %) What design pattern can you use in method AudioProcessor::ParallelCompress() for instantiating the threading classes of aos and bsystem without depending on the concrete classes in the method?

(b) (7 %) Please provide the class diagram of the design and also show the reimplemented method in c++.

3. Now you are supporting another platform, cplatform, that provides the following threading classes:



It can be seen that the cplatform threading classes provide similar functions with an incompatible interface. You need extra work to make the method support cplatform.

(a) (3 %) Which design pattern can you use to make AudioProcessor::ParallelCompress() use the cplatform classes without depending on its interface?

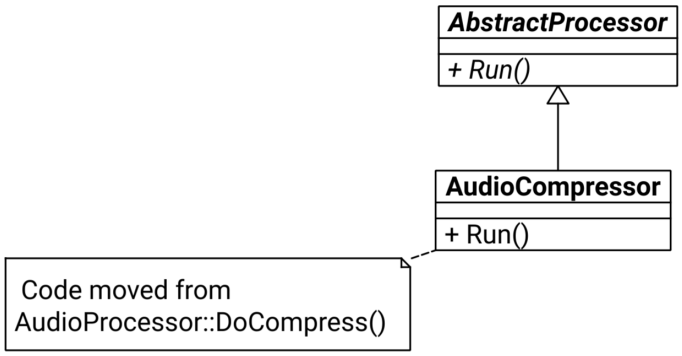
(b) (5 %) Please provide the design in a class diagram.

4. The capability of running computation-intensive workloads on separate threads is useful not only for audio compression but also for other audio processing functions like removing background noise, echo cancellation, speech recognition, etc. You renamed the method and parameterized it with an AbstractProcessor argument:

```
1 using namespace std;
2 void AudioProcessor::ParallelProcess(
3     const vector<AudioFile>& files,
4     int nThreads,
5     AbstractProcessor* processor) {
6   vector<unique_ptr<aos::Thread>> threads(
7       nThreads);
8   // Create a aos::MutexLock instance.
```

```
 9    unique_ptr<aos::MutexLock> lock(
10        new aos::MutexLock());
11
12    int fIndex = files.size();
13    for (int i = 0; i < nThreads; ++i) {
14      // Create a aos::Thread instance.
15      threads[i].reset(new aos::Thread());
16
17      // Run the lambda on a thread.
18      threads[i]->Start([&](){
19        while (true) {
20          // Get an unprocessed AudioFile instance.
21          lock->Lock();
22          if (fIndex == files.size()) {
23            // Returning from the lambda will
24            // terminate the thread.
25            return;
26          }
27          const auto& file = files[fIndex++];
28          lock->Unlock();
29
30          // Run the processor with the AudioFile
31          //instance.
32          processor->Run(file);
33        }
34      });
35    }
36
37     // Wait until all threads are terminated.
38    for (auto& thread : threads) {
39      thread->Join();
40    }
41  }
```

The original AudioProcessor::DoCompress() method is moved into AudioCompress::Run():



(a) (3 %) Which pattern is used to refactor the design to make it support generic processor implementations?

(b) (4 %) Please add the support of EchoCancellation in a class diagram.

5. (10 %) Why do we need both verification and validation in an adequate software development process? Please explain the reasons using CONCRETE EXAMPLES.

6. (20 %) Construct an abstract data model for the new room-booking system of a large hotel that has to meet the following requirements:

   - The hotel has several thousands of rooms, classified into several types, including single, double, suite, etc., which might change slightly over time.
   - Every room has a unique number and may be remodeled into a different type.
   - According to statistics, nearly all past customers prefer non-smoking rooms and many does not like rooms with pet lingering smell; these tendencies are expected to continue.
   - Rooms are priced according to their types and days of the week normally, but the rates may vary for holidays.
   - A customer making reservations must leave her/his full name and email address or phone number.
   - A customer may reserve several rooms, each for multiple days.
   - For promotional purposes, the hotel decides to implement a membership program; members may enjoy discounted room rates.

   You should avoid many-to-many relationships; otherwise, you must provide a verbal explanation for including such relationships. Please use the UML as much as possible when describing the model. State the assumptions, if any, you make for your construction.

7. (5 %) People may use a same term, but actually mean different things. The same person may also use a term to mean different things in different contexts. The following two statements appear to be contradictory.

   - A black-box testing can never be static.
   - Testing/examining the specification is a static black-box testing.

   Please explain in what sense these two statements can both be true.

8. (10 %) We have considered in class an example (two signs at the entrance of an escalator) showing ambiguity of natural language (English). Please give another example in English or Chinese and show how logic formulae may help clarify/reveal the ambiguity. The example can be of one sentence having two different meanings or two sentences with the same syntactic structure but different meanings.

9. Please provide a precise description, using logical formulae, for each of the following requirements. The functions/constants and predicates you may use are: $+$, $-$, 0, 1, $<$, $=$, $\leq$, plus those introduced in the requirement statements. Make assumptions where you see necessary.

   (a) (5 %) The returned result `r` is correct for a search function that is supposed to find the LAST occurrence of the value of target `t` in an array $A[0..N-1]$ of integers; we stipulate that `r` $= -1$ if the value of `t` cannot be found in A.

   (b) (5 %) The array $B[0..N-1]$ (of integers) represents a correct partition, as in the quick sort, of array $A[0..N-1]$, with $A[0]$ used as the pivot. The quick sort eventually would arrange the entries of A in an increasing order.