Project 1 Frequently Asked Questions

1. How do I compile nachos with preprocessor directives?

Once you have defined your tags and wrapped your code with them, e.g. HW1_OFFICE, HW1_SEMAPHORES, etc. you will have to compile nachos with those flags in order to see your output.

Threre are 2 ways of doing that:

1. In the threads directory there is a file called 'Makefile'. in there you will see a line that reads:
DEFINES= -DTHREADS
you will have to modify this line to read something like this:
DEFINES= -DTHREADS -DHW1_SEMAPHORES -DBLAH ..... and so on.
Remember: only include the ones you want to run.

2. If you are using the Bash shell (type 'bash'), you can do it in one line, without modifying any files by using a single command. Here is the list of them (**NOTE**: Do NOT type the "$" sign):

Default behavior:
$ export DEFINES="-DTHREADS"; rm *.o; make; ./nachos

To test threads with semaphores:
$ export DEFINES="-DTHREADS -DHW1_SEMAPHORES"; rm *.o; make; ./nachos

To test threads with locks:
$ export DEFINES="-DTHREADS -DHW1_LOCKS"; rm *.o; make; ./nachos

To test the elevator solution:
$ export DEFINES="-DHW1_ELEVATOR"; rm *.o; make; ./nachos

2. How do I run nachos with the -rs option?

You have to specify a random seed with the rs option as:

./nachos -rs 43 .... (here 43 is the random seed.)

3. What are these core files and where do they come from?

Core files are generated when your program "Crashes". they can be quite large and take up a lot of space over time. Please make sure you have deleted all core files that you may have generated before you submit your assignment. These take a lot of space. One way to know quickly if there is a core file in your directory structure is to look at the size of the compressed archive.

4. I have created myFunction(int i) but it says it's not declared. what do I do?

Unlike Java, C++ needs a function signature, or prototype declared for a function to work. In this case, if you have:

```
myFunction(int i) {
printf("Hello World");
}
```

you will also need:

```
myFunction(int i);
```

declared somewhere before the function call. This signature or prototype is usually in a header file (myClass.h) but sometimes also in the implementation file (myClass.cc). Look through some of the source code of nachos and you will find plenty of examples.

5. I am having problems creating and using objects in my code. what's wrong?

If you follow the coding guidelines in nachos and base yourself on examples of code already written, you shouldn't have any problems. Unfortunately, there are many ways to do the same thing and one of the common problems is creating and using objects.

Here are 2 different ways of doing this:

This is the "Official" way you see in nachos:
```
Semaphore * sem;
...
sem = new Semaphore("mySem", 1);
...
sem->P();
sem->V();
```

Threre is also this way:
```
extern Semaphore sem;
...
Semaphore sem("mySem",1);
...
sem.P();
sem.V();
```

Both of them work, so you should be able to use any of them. Just make sure your code is consistent. Don't go back and forth, it's just confusing.

6. *What is the expected output for Exercise 1?*

Suppose you set parameter to 4, your program should create threads 0,1,2,3. Also, when you use the semaphore to synchronize the SharedVariable, the value printed out should be strictly incremented. And the final value should be 5*<number of threads>. A sample output for 2 threads should be

```
......
Thread 0 sees the value 0
Thread 1 sees the value 1
Thread 0 sees the value 2
Thread 1 sees the value 3
Thread 0 sees the value 4
Thread 1 sees the value 5
Thread 0 sees the value 6
Thread 1 sees the value 7
Thread 0 sees the value 8
Thread 1 sees the value 9
……
```

Thread 1 sees final value 10
Thread 0 sees final value 10

7. *In exercise 1, should the thread 0 print the "see the final value" first? How can we implement barrier?*

Barrier makes sure each thread will not continue until all threads achieve the specific point (i.e., finish 5 iterations of updating the SharedVariable in this case). It does not matter which thread print out the "see the final value". However, all "see the final value" should be print out at the very end of your program.
You can think of using the P() and V() operation to implement the "barrier".

8. *For Exercise 2 does the locks solution need to also work when random interrupts are turned on using the -rs option or is this limited to exercise 1 with semaphores?*

All the exercises should be working with -rs option.

9. *Can I remove the currentThread->Yield() statements in SimpleThread() when using semaphores?*

No. currentThread->Yield() and –rs option will mass up your updating of global variable SharedVariable if you cannot use semaphore correctly. Your program should synchronize correctly even with the presence of currentThread->Yield() and –rs option.

10. *How to verify the correctness of Lock and Condition in exercise 2 and 3 ?*

Replace the semaphore you used in the exercise 1 with lock or condition, to verify if the synchronization runs correctly. You can also use the user programs in the link for Project 1 resources to test your implementations for locks, conditions and semaphores.

7. How many students/faculties is enough to test exercise 4?

The number of students/faculties should be big enough (more than 20) to proof the correctness of multi-thread program.