True or False. Answer the questions below.

1. (True or False) Without proper synchronization, global variables are unsafe to write to by multiple threads. However, local variables (e.g. variables declared inside of a function) are always safe to write from multiple threads.

2. (True or False) Threads do not require help from the OS to communicate with the other threads in the same process.

3. (True or False) When mixing threads with processes, it is better to first call `fork`, then create the threads rather than create the threads first, and then call `fork`.

4. (True or False) A multi-threaded process will always complete its work faster than a single-threaded process.

5. (True or False) Communication between processes is generally more efficient than communicating between threads.

6. (True or False) If one thread of a multithreaded process crashes (e.g. divides by 0), the entire process will be terminated, including all other threads.

7. (True or False) A multithreaded process can only run on a computer with multiple CPUs or multiple cores.

8. (True or False) When multiple threads are created in a process, the last thread created will always be the last thread to run.

9. (True or False) It is easier to share pointers between threads than it is to share them between processes.

Multiple Choice. Choose the letter of the best answer.

10. What does the `printf` statement below print out? (Assume that all calls succeed.)

```
int foo = 0;

void* Inc(void *p)
{
  foo++;
  return NULL;
}

int main(void)
{
  pthread_t tid1, tid2;
  pthread_create(&tid1, NULL, Inc, NULL);
  printf("%i\n", foo);
  pthread_create(&tid2, NULL, Inc, NULL);
  return 0;
}
```

(a) 0

(b) 1

(c) 2

(d) The result is undefined.

Short Answer. Answer the questions below.

11. A _____ allows you to re-uses an existing thread rather than creating a new one.

12. All threads in a process share global memory but they have their own _____ and _____.

13. Always having many threads in the _____ state means that there are not enough CPU cores to handle them.

14. Suppose a thread (main thread) creates 3 worker threads, T1, T2, and T3, and they are all running on a quad-core CPU. If the times to complete their tasks are about 1, 3, and 2 seconds, respectively, about how long will the main thread need to wait for all worker threads to finish? Assume no other threads are running.

15. A thread is sometimes called a light-weight process or a _____ process.

16. In the Pthreads library, what is the name of the function that is called when one thread wants to wait on another thread until it finishes?

17. In a multi-tasking operating system, if a thread, `T1`, is in the middle of calculating digits of `pi` and needs to give up the CPU for another thread, what state will `T1` be put in?

18. What is the prototype (declaration) of a thread function in the Pthreads library? Assume the name of the function is `ThreadFn`.

19. Given the code below, what are the values of the PIDs printed on lines A, B, C, and D? (Assume that all calls succeed and the actual PIDs are: parent PID is 100, and child PID is 500.)

```
int main(void)
{
  int pid1;
  int pid2 = fork();
  if (pid2 > 0)
  {
    pid1 = getpid();
    printf("%i", pid1); /* A */      A. pid1 is _____________
    printf("%i", pid2); /* B */      B. pid2 is _____________
  }
  else
  {
    pid1 = getpid();
    printf("%i", pid1); /* C */      C. pid1 is _____________
    printf("%i", pid2); /* D */      D. pid2 is _____________
    wait(NULL);
  }
  return 0;
}
```

20. There are several states that a thread can be in, although it can only be in one state at any one time. If a thread is in the **running** state, what are the other possible states that the thread can transition to?

21. The ability to specify on which CPU/core a thread/process can execute is called _____.

22. Look closely at the code below. How many times is the word **forked** printed to the screen? Assume that all of the calls to **fork** succeed.

```
int main(void)
{
  if ( fork() || fork() )
    fork();

  printf("forked\n");
  return 0;
}
```

23. Why did we NOT use the `pthread_exit` function in our programs in CS180?

24. Programs running under Linux can have their `stdout` output redirected to a file using the greater than symbol (`>`). What command do you use to have `stdout` redirected to a file but also have the output displayed on the screen?

25. What happens when you type `cd ~foobar` from the bash shell prompt? Be specific.