1. A process control block ____.
A) includes information on the process's state
B) stores the address of the next instruction to be processed by a different process
C) determines which process is to be executed next
D) is an example of a process queue

2. The _____ refers to the number of processes in memory.
A) process count
B) long-term scheduler
C) degree of multiprogramming
D) CPU scheduler

3. When a child process is created, which of the following is a possibility in terms of the execution or address space of the child process?
A) The child process runs concurrently with the parent.
B) The child process has a new program loaded into it.
C) The child is a duplicate of the parent.
D) All of the above

4. A process may transition to the Ready state by which of the following actions?
A) Completion of an I/O event
B) Awaiting its turn on the CPU
C) Newly-admitted process
D) All of the above

5. A blocking `send()` and blocking `receive()` is known as a(n) _____
A) synchronized message
B) rendezvous
C) blocked message
D) asynchronous message

6. Which of the following statements is true?
A) Shared memory is typically faster than message passing.
B) Message passing is typically faster than shared memory.
C) Message passing is most useful for exchanging large amounts of data.
D) Shared memory is far more common in operating systems than message passing.

True/False Questions

7. All processes in UNIX first translate to a zombie process upon termination.
8. The `exec()` system call creates a new process.
9. For a single-processor system, there will never be more than one process in the Running state.
10. Shared memory is a more appropriate IPC mechanism than message passing for distributed systems.

11. Including the parent process, how many processes are created by the following code? Please draw the process tree.

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    /* fork a child process */
    fork();

    /* fork another child process */
    fork();

    /* and fork another */
    fork();

    return 0;
}
```