COMPSCI 3SH3 MIDTERM 1

1. What one of the following is not true?

a) Kernel is the program that constitutes the central core of the operating system
b) Kernel is the first part of operating system to load into memory during booting
c) Kernel is made of various modules which cannot be loaded in running operating system
d) Kernel remains in the memory during the entire computer session

Answer: (C)

2. The OS structure used by the original UNIX is monolithic. The reason this architecture is used is:

a) Easy to implement and extend
b) Speed and efficiency
c) It is more flexible
d) It consumes less power to execute an instruction

Answer: (B)

3. Darwin OS is a layered system that consists primarily of:

a) Linux kernel and UNIX kernel
b) Android kernel
c) Linux kernel and Mach microkernel
d) Mach microkernel and BSD UNIX kernel

Answer: (D)

4. How are iOS and Android similar?

a) Both are based on existing kernels (Linux and macOS)
b) Both are open source
c) Both use virtual machine
d) iOS and Android applications are developed in Python

Answer: (A)

5. Interrupt response time is usually shorter in polling system than in vectored interrupt system. (T/F)

Answer: (F)

6. Interrupts are synchronous events. (T/F)

Answer: (T) & (F) – Depends on interpretation of the statement

7. Primary storage is a volatile type of memory. (T/F)

Answer: (T)

8. SSD is primary storage. (T/F)

Answer: (F)

9. Non-volatile memories are faster than volatile memories. (T/F)

Answer: (T)

10. Asymmetric multi-processing is a subset of symmetric multi-processing. (T/F)

Answer: (F)

11. It is not possible to have incoherent caches in dual-core multi-processor architecture. (T/F)

Answer: (T)

12. Cache is managed by the OS. (T/F)

Answer: (F)

13. Rank the following storage mediums from fastest to slowest:

    1. Hard-disk drives
    2. Registers
    3. Optical disk
    4. Main memory
    5. Non-volatile memory
    6. Magnetic tapes
    7. Cache

Answer: Registers, Cache, Main Memory, Nonvolatile memory, Hard-disk drives, Optical disk, Magnetic tapes

14.

a) How many times "hello" is printed by the program shown below?

```
// Listing 1: Processes Creation
#include <stdio.h>
#include <unistd.h>

int main() {
```

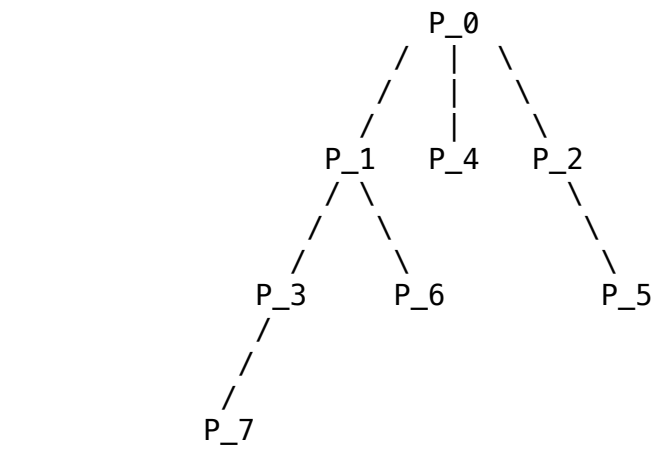
```
        fork();
        fork();
        fork();
        printf("hello\n");

        return 0;
    }
```

Answer: 8 Times

b) Represent the relationship between the processes as a TREE hierarchy. The main process is P0, and the processes created by forks P1...P(n-1), where `n` is the number of processes.

Answer:



The main process: P0
Processes created by the 1st fork: P1
Processes created by the 2nd fork: P2, P3
Processes created by the 3rd fork: P4, P5, P6, P7

15.

a) Write the consumer code/process that consumes Collatz sequence created by the producer processes shown below.

```
// Listing 2: Producer
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/types.h>
```

```
   int main(int argc, char * argv[]) {

       const int SIZE = 2048;
       const char * name = "SeqCOLLATZ";

       int shm_fd;
       void * ptr;
       char str[128];
       int n = atoi(argv[1]);

       shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);

       ftruncate(shm_fd, SIZE);

       ptr = mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED,
                  shm_fd, 0);

       if (ptr == MAP_FAILED) {
           printf("Map failed\n");
           exit(-1);
       }

       sprintf(str, "%d, ", n);
       sprintf(ptr, "%s ", str);
       while (n != 1) {
           ptr += strlen(str);

           if (n % 2 == 0) {
               n = n / 2;
           } else {
               n = (3 * n) + 1;
           }
           sprintf(str, "%d, ", n);
           sprintf(ptr, "%s", str);
       }

   return 0;

   }
```

Answer:

```
// Listing 3: Consumer

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
```

```
int main() {

    const char * name = "SeqCOLLATZ";
    const int SIZE = 2048;

    int shm_fd;
    void * ptr;
    int i;

    /* Open the shared memory segment */
    shm_fd = shm_open(name, O_RDONLY, 0666);
    if (shm_fd == -1) {
        printf("Shared memory failed\n");
        exit(-1);
    }

    /* Map the shared memory segment in the address space of the
     * process
     */
    ptr = mmap(0, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
    if (ptr == MAP_FAILED) {
        printf("Map failed\n");
        exit(-1);
    }

    /* Read from the shared memory region */
    printf("%s", (char *) ptr);

    /* Remove the shared memory segment */
    if (shm_unlink(name) == -1) {
        printf("Error removing %s\n", name);
        exit(-1);
    }

    return 0;
}
```