Solve the following problems

1. Consider the following figure ……

- Including the initial parent process,

-How many processes are created by the program?

- What statement(s) you can add to check if fork() failed and an error occurred?

```
int main()
{
    int i;

    for (i = 0; i < 4; i++)
      fork();

    return 0;
}
```

---

2. **A.** Assume you want to implement a web-server for **YouTube** by using multithreading, where each thread serves one incoming request by loading a video file from the disk. Assume the OS only provides the normal blocking read system call for disk reads, do you think user-level threads or kernel-level threads should be used? Why?

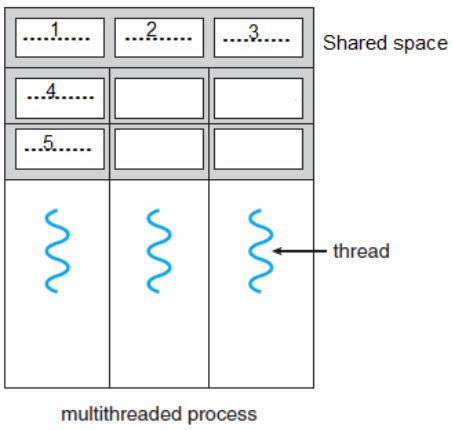
B. Now you want to implement a web-server for **Facebook**, to serve each user's "Home" page (the first page you see after you log in). This time your web-server needs to perform many tasks: load the news feeds from each of your friends, load the advertisement, check for new messages, etc. Now you want to implement your web-server by using multithreading, and have one thread to perform each of the tasks, and later these threads will cooperate with each other to collectively construct the "Home" page. For performance reasons, Facebook makes sure that all the data these threads need is already cached in the memory (so they don't have to perform any disk I/O). Do you think user-level threads or kernel-level threads should be used? Why?

---

3. Consider the following figure

a- complete the missing parts,
show which parts are shared by all threads and which are not?

b- What's the difference between a process starting another copy of itself and starting another thread?



---

4. Recall that protected instructions can only be executed in the kernel mode (i.e., can only be executed by the OS but not by user-level processes). For each instruction below, is it a protected instruction? ***(No explanations needed).***
(A) load instruction (read a value from memory into a register)

(B) modify the PC register (program counter)
(C) modify the SP register (stack pointer)
(D) modify the register that controls kernel/user mode
(E) direct access I/O device

---

5. Write a shell script that accepts a file name, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

---

6. Assume that you have a page-reference string for a process with m frames (initially all empty). The page-reference string has length p; n distinct page numbers occur in it. Answer these questions for any page-replacement algorithms:
a. What is a lower bound on the number of page faults?
b. What is an upper bound on the number of page faults?

---

7. Consider the following page reference string:
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.
How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.
• LRU replacement
• FIFO replacement
• Optimal replacement

---

8. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use non-preemptive scheduling, and base all decisions on the information you have at the time the decision must be made. (Draw Gantt chart for process sequence)

Process	Arrival Time	Burst Time
*P*1	0.0	8
*P*2	0.4	4
*P*3	1.0	1

a. What is the average waiting time for these processes with the FCFS scheduling algorithm?
b. What is the average waiting time for these processes with the SJF scheduling algorithm?