1. **(20 points)** Explain what are preemptive scheduling and non-preemptive scheduling? What are design tradeoffs between preemptive scheduling and non-preemptive scheduling.

2. **(30 points)** Suppose that processes with the following CPU bursts (4, 3, 5, 7, 1, 9) have arrived in the ready queue in the order given at time 0;

No other processes enter the system. Draw a Gantt chart and calculate the average waiting times for each of the following scheduling algorithms. Show your waiting time calculation.
   a. FCFS (First Come First Served)
   b. SJF (Shortest Job First, non-preemptive)
   c. RR (Round Robin) with a quantum of 2

3. **(30 points)** Suppose that three processes A, B and C with CPU time 6, 1 and 4 (respectively) arrive at time 0, and process D with CPU time 2 arrives at time 2, and process E with CPU time 3 arrives at time 9. Draw the Gantt chart of the schedules and calculate the average waiting times for the following scheduling algorithms:
   a. SJF (Shortest Job First, non-preemptive)
   b. SRJF (Shortest remaining time job first, preemptive)

4. **(30 points)** Writing two programs, producer.c and consumer.c, and these two programs are using the shared memory to communicate. Producer is writing a string “Hihi, Producer” to the same memory segment. Consumer is going to read from this memory segment and print the result. In the end of consumer, **you should remove** the created shared memory segment in the system in your code. You can confirm this by using “ipcs -m”.

Tips: To create a memory segment connecting with two processes, you may use the following API:

```
key_t ftok(const char *pathname, int proj_id);
```

Note: Provide the code in your handout. You can remove all those header files to save space. I suggest you to actually run this program on the system.