Assigned reading: Hoover, Chapter 5 and 6, Chapter 7, Sections 1 through 4. Each problem is worth 10 points

From Chapter 5, starting on page 164
1. Number 1
2. Number 2
3. Number 7
4. Number 8
5. Number 12 (see the **devterm.c** files from Chapter 5 lecture notes). We will use a similar approach in our next two machine problems.

From Chapter 7, starting on page 251
6. Number 2
7. Number 4
8. Number 6
9. Give the output for the following code. (Recall the **wait()** function pauses until any child process has completed. If a process does not have a child the function returns immediately.)

a) With the **wait(NULL)** line commented out:
b) With the **wait(NULL)** line not commented out:

```
int main(void)
{
  int i = 5, pid;
  printf("1:%d\n", i);
  pid = fork();
  i++;      printf("2:%d\n", i);

//  wait(NULL);

  if (pid == 0) {
    sleep(1);
    i--;     printf("3:%d\n", i);
    sleep(1);
    i += 4; printf("4:%d\n", i);
  } else {
    i++;     printf("5:%d\n", i);
    i = i+3;
    printf("6:%d\n", i);
    sleep(6);
    i = 0;  printf("7:%d\n", i);
  }
  i += 2;    printf("8:%d\n", i);
}
```

10. Write a program using signals that accepts two commands from the keyboard: **poke** or **quit**. When the program begins it waits for input from the user. When the user types **poke**, begin to print "**Go Tigers**" to standard output and repeat the message once each second. When the user types **poke** again the prints should stop and there is no output. Each time the user enters **poke** toggle the

printing status between on and off. When the user types **quit**, print "**Beat SCar**" and exit the program. Invalid input should be ignored. No output is required for invalid input or after **quit** is entered.

Turn in a paper copy of your solutions in class. Do not submit electronically. While we have a policy for late submission of programming assignments, late submission of homework assignments will not be accepted.