CSE 5/7343-Operating Systems
Program 1
Due: Monday, February 12 by 11:59pm.

Using the Unix OS environment, write a c program named **sleepy** that gets a loop count from the command line as in

```
sleepy n
```

where **n** is the number of seconds for which the program should run. Implement this timing by using a loop that iterates **n** times. The body of the loop will contain the statement **sleep(1)**. This will put the program to sleep for one second **n** times before exiting. If **n** is not provided use a default value of 5. Here is the code needed to pass an argument to a program from the command line:

```
int main(int argc, char *argv[ ]) {
    int count;
    if (argc == 2)
        count = atoi(argv[1]);
    else
        count = 5;
}
```

```
#include <unistd.h>
...
unsigned int sleep(unsigned int seconds);
```

This function causes the calling process to be suspended until either

1. the amount of wall clock time specified by **seconds** has elapsed, or
2. a signal is caught by the process and the signal handler returns.

The return value from **sleep** is 0 or the number of unslept seconds if the sleep function has been interrupted by a signal interrupt. You may assume that the program always sleeps the number of seconds requested.

In each iteration print out the process ID and the loop count so that that particular process can be identified.

The process ID can be obtained from the **getpid** function:

```
#include <sys/types.h>
#include <unistd.h>
...
pid_t getpid(void);
```

This function returns the process ID of the calling process.

The process ID is returned as a type **pid_t** which is actually an integer so it can be treated as such in a **printf** format statement (use an **int** cast to avoid a compile warning from **gcc**).

Code should be in 'straight' c using the compiler of your choice (**cc** or **gcc**). Using the –o compiler option will allow you override the default a.out executable name.

Here is an example run of sleepy.

```
>sleepy 4
 176987-START
 176987-TICK 1
 176987-TICK 2
 176987-TICK 3
 176987-TICK 4
```

If you do not wish to use the –o compiler option to override the default a.out executable name an example run might look like this:

```
>./a.out 4
    176987-START
    176987-TICK 1
    176987-TICK 2
    176987-TICK 3
    176987-TICK 4
```

Run your program several times. Does it always have the same process id? What state do you think the process transitions to after a call to the sleep function? Include your answers as comments in your source code.

You only need to submit you source code when you turn in the program. Using in-line comments indicate which server (genuse26, genuse27, etc.), c compiler and compiler options you used to develop your program. Submit your program via Canvas. Make sure you test you program on one of the genuse servers before turning it in.