A *student* approaches the *Dean* of the College to take a science class. The Dean assigns a *prof*, who is smart and capable, but is also very tired. If the professor falls asleep while giving the lecture then the student will wait for NUM_SNORES_BEFORE_STUDENT_COMPLAINS snores before waking the professor. On the NUM_COMPLAINTS_TO_PROF_BEFORE_COMPLAIN_TO_DEAN-th attempt the student bypasses the prof and goes straight to the dean, who also yells at the prof.

1. **Copy and paste the following 3 files:**

```
/*-------------------------------------------------------------------------*
 *---                                                               ---*
 *---         sleepyProfHeaders.h                                   ---*
 *---                                                               ---*
 *-------   ----    ----    ----    ----    ----    ----       ---*
 *---                                                               ---*
 *---Version 1a                                                     ---*
 *---                                                               ---*
 *-------------------------------------------------------------------------*/

#include    <stdlib.h>
#include    <stdio.h>
#include    <string.h>
#include    <signal.h>
#include    <unistd.h>
#include    <sys/types.h>
#include    <sys/wait.h>


//  PURPOSE:  To tell the length string arrays.
#define          MAX_LINE                          64

//  PURPOSE:  To tell the name of the professor.
#define          PROFS_NAME                        "Prof
Narcolepsy"

//  PURPOSE:  To tell the name of the program the professor process should
run.
#define          PROF_PROGRAM                      "prof"

//  PURPOSE:  To tell the name of the program the student process should
run.
#define          STUDENT_PROGRAM                   "student"


//  PURPOSE:  To help determine how long a class period is.
#define          NUM_MOMENTS_IN_CLASS              90


//  PURPOSE:  To help determine how long a class period is.
#define          SECONDS_PER_MOMENT                1

//  PURPOSE:  To tell the probability that the professor will fall asleep
in a
//   moment.
#define          PROF_FALL_ASLEEP_PROB             0.125
```

```c
//   PURPOSE:   To tell the number of sleeping moments that must occur before
the
//     student complains.
#define             NUM_SNORES_BEFORE_STUDENT_COMPLAINS    3


//   PURPOSE:   To tell the number of complaints to the professor that must
occur
//     before the student complains to the dean.
#define             NUM_COMPLAINTS_TO_PROF_BEFORE_COMPLAIN_TO_DEAN        \
                                                  3

//   PURPOSE:   What the professor sends to the student to teach him or her.
#define             PROF_TEACH_SIGNAL                       SIGUSR1


//   PURPOSE:   What the professor sends to the student to say that he or she
is
//     sleeping.
#define             PROF_SNORE_SIGNAL                       SIGUSR2


//   PURPOSE:   What the student sends to the professor and dean to complain
about
//     the sleeping prof.
#define             COMPLAIN_SIGNAL                         SIGUSR1


//   PURPOSE:   What the dean sends to the professor, and the professor to
the
//     student, to say the class is over.
#define             CLASS_DISMISSED_SIGNAL                  SIGINT


//   PURPOSE:   To distinguish among the academic topics the professor can
teach.
typedef             enum
            {
              NO_SUBJECT,
              PHYSICS_SUBJECT,
              CHEMISTRY_SUBJECT,
              BIOLOGY_SUBJECT,

              MIN_SUBJECT   = PHYSICS_SUBJECT,
              MAX_SUBJECT   = BIOLOGY_SUBJECT,

              NUM_SUBJECTS
            }
            subject_ty;


//   PURPOSE:   To hold the names of the academic subjects references by
//     'subject_ty'.
extern
const char* subjectName[NUM_SUBJECTS];
```

```c
//  PURPOSE:  To return the corresponding 'subject_ty' value of the subject
//   named 'subjectNameCPtr', or to return 'NO_SUBJECT' if no match is
found.
extern
subject_ty  getSubjectFromName      (const char*    subjectNameCPtr
                                     );


/*-------------------------------------------------------------------------*
 *---                                                                   ---*
 *---          dean.c                                                   ---*
 *---                                                                   ---*
 *-------      ----     ----     ----     ----     ----     ----     -------*
 *---                                                                   ---*
 *---Version 1a                                                         ---*
 *---                                                                   ---*
 *-------------------------------------------------------------------------*/

#include     "sleepyProfHeaders.h"

//  PURPOSE:  To hold the process id of the professor (child) process.
pid_t              profPid;

//  PURPOSE:  To hold '1' while the professor process is running.
int        isProfRunning  = 0;


//  PURPOSE:  To ask the user which subject they want to study, and to
return
//   the corresponding 'subject_ty' value.  No parameters.
subject_ty  getSubject ()
{
 int i;
 int subject;
 char       text[MAX_LINE];

 printf("Dean \"Welcome to the Bubu University!\"\n");

 do
 {
  printf("Dean \"What would you like to study?\"\n");

  for  (i = (int)MIN_SUBJECT;  i <= (int)MAX_SUBJECT;  i++)
  {
   printf("  %d: %s\n",i,subjectName[i]);
  }

  fgets(text,MAX_LINE,stdin);
  subject = strtol(text,NULL,0);
 }
 while  ( (subject < MIN_SUBJECT) || (subject > MAX_SUBJECT) );

 return((subject_ty)subject);
}
```

```c
//  PURPOSE:  To tell the professor to wake up.  Ignores 'sigNum'.  No
return
//    value.
void        complainSigHandler
                                (int     sigNum
                                )
{
 printf("Dean \"WAKE UP!\"\n");
 kill(profPid,COMPLAIN_SIGNAL);
}


//  PURPOSE:  To handle when the professor process finishes.  Ignores
'sigNum'.
//    No return value.
void        sigChildHandler(int     sigNum
                                )
{
 int status;

 wait(&status);

 if  ( WIFEXITED(status) && (WEXITSTATUS(status) == EXIT_SUCCESS) )
 {
  printf("(The Dean turns of the lights and locks the classroom)\n");
 }
 else
 {
  printf("Dean \"Whoops!  I will look into that.\"\n");
 }

 isProfRunning = 0;
}


//  PURPOSE:  To install 'complainSigHandler()' as the signal handler to
handle
//    'COMPLAIN_SIGNAL' coming to the dean.  No parameters.  No return
value.
void        installSignalHandlers
                                ()
{
 struct sigaction   act;

 memset(&act,'\0',sizeof(act));
 act.sa_handler     = complainSigHandler;
 sigaction(COMPLAIN_SIGNAL,&act,NULL);

 act.sa_handler     = sigChildHandler;
 sigaction(SIGCHLD,&act,NULL);
}


//  PURPOSE:  To start the professor process (and initialize 'profPid'),
and
//    to have the professor process teach 'subject'
void        startProf       (subject_ty     subject
```

```
                                    )
{
 char           text[MAX_LINE];

 printf("Dean: \"%s, today you will teach %s\"\n",
        PROFS_NAME,subjectName[subject]
       );

 isProfRunning        = 1;
 profPid     = fork();

 if  (profPid == 0)
 {
  execl(PROF_PROGRAM,PROF_PROGRAM,subjectName[subject],NULL);
  printf("Prof \"I cannot find my classroom!  I quit!\"\n");
  exit(EXIT_FAILURE);
 }


}


//  PURPOSE:  To wait for either the class to be over (more specifically,
for
//  'NUM_MOMENTS_IN_CLASS * SECONDS_PER_MOMENT' seconds to elapse), or
//  to be notified that the professor process no longer runs.  No
//  parameters.  No return value.
void        waitUntilClassPeriodIsOver
                                       ()
{
 int momentNum;

 for  ( momentNum = 0;  momentNum < NUM_MOMENTS_IN_CLASS;  momentNum++ )
 {
  sleep(SECONDS_PER_MOMENT);

  if  (!isProfRunning)
  {
   break;
  }

  if  ( (momentNum % 5) == 0 )
  {
   printf("(The Dean shuffles paper.)\n");
  }
 }

}


//  PURPOSE:  To inform the professor process that class is over, and to
wait
//  to be notified by the OS that it has finished.  No parameters.  No
//  return value.
void        informProfClassIsOver  ()
{
 printf("Dean \"Class is over!\"\n");
```

```
 kill(profPid,CLASS_DISMISSED_SIGNAL);

 while  (isProfRunning)
  sleep(1);
}


//  PURPOSE:  To run the program.  Ignores command line arguments.  Returns
//   'EXIT_SUCCESS' to OS.
int       main            ()
{
 int subject = getSubject();

 installSignalHandlers();
 startProf(subject);
 waitUntilClassPeriodIsOver();

 if  (isProfRunning)
 {
  informProfClassIsOver();
 }

 return(EXIT_SUCCESS);
}


/*------------------------------------------------------------------*
 *---                                                            ---*
 *---          subjectName.c                                     ---*
 *---                                                            ---*
 *-------    ----   ----   ----   ----   ----   ----   ------*
 *---                                                            ---*
 *---Version 1a                                                  ---*
 *---                                                            ---*
  *------------------------------------------------------------------*/


#include    "sleepyProfHeaders.h"


//  PURPOSE:  To hold the names of the academic subjects references by
//   'subject_ty'.
const char* subjectName[NUM_SUBJECTS]              =
            {
              "",
              "Physics",
              "Chemistry",
              "Biology"
            };


//  PURPOSE:  To return the corresponding 'subject_ty' value of the subject
//   named 'subjectNameCPtr', or to return 'NO_SUBJECT' if no match is
found.
subject_ty  getSubjectFromName      (const char*    subjectNameCPtr
                                    )
{
 int i;
```

```
 for  (i = (int)MIN_SUBJECT;  i <= (int)MAX_SUBJECT;  i++)
 {
  if  (strcmp(subjectNameCPtr,subjectName[i]) == 0)
  {
   return((subject_ty)i);
  }
 }

 return(NO_SUBJECT);
}
```

2. Copy, paste, and finish `prof.c`:

```
/*-------------------------------------------------------------------------*
 *---                                                                   ---*
 *---        prof.c                                                     ---*
 *---                                                                   ---*
 *------- ----     ----     ----     ----     ----     ----     ------- *
 *---                                                                   ---*
 *---Version 1a                                                         ---*
 *---                                                                   ---*
 *-----------------------------------------------------------------------*/
```

```
#include    "sleepyProfHeaders.h"


//  PURPOSE:  To hold '1' while class still is in session, or '0' after
class
//   is over.
int         isStillClassTime        = 1;


//  PURPOSE:  To hold '2' when the prof is quite awake, '1' when the prof
is
//   about to fall asleep, or '0' when the prof is asleep.
int         awakeLevel              = 2;


//  PURPOSE:  To hold the process id number of the student (child) process.
pid_t               studentPid;


//  PURPOSE:  To tell how many facts are known for each subject.
#define             NUM_FACTS_PER_SUBJECT  4


//  PURPOSE:  To tell the physics facts.
const char* PHYSICS_KNOWLEDGE[NUM_FACTS_PER_SUBJECT]     =
            { "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt"
              " when dm/dt = 0.",

              "Fermions have 1/2 spin and cannot occupy the same quantum"
              " state, Bosons have integer spin and can occupy the same"
              " quantum state.",
```

```cpp
                 "The electron-electron repulsion between atoms and
molecules"
                 " supports the structure of most matter at our scale. If "
                 "gravity overcomes this electron repulsion, then electrons "
                 "collapse into the nucleus and matter becomes a neutron "
                 "star.",

                 "There is a large Black hole at the center of our galaxy."
              };

//  PURPOSE:  To tell the chemistry facts.
const char*  CHEMISTRY_KNOWLEDGE[NUM_FACTS_PER_SUBJECT]    =
              {"In SN2 reactions, the nucleophile puts electron density
into"
                 " the anti-bonding orbital of leaving group, thus weakening"
                 " the bond between the leaving group and the substrate.",

                 "The transition state of a reaction is the configuration of "
                 "highest energy.",

                 "The energy difference between the reactants and the "
                 "transition state determines the rate of reaction",
```

```cpp
                 "The Diels Alder reaction is a popular way to make cyclic "
                 "compounds with atoms other than carbon."
              };

//  PURPOSE:  To tell the biology facts.
const char*  BIOLOGY_KNOWLEDGE[NUM_FACTS_PER_SUBJECT]      =
              {"Allopatric speciation happens when some geological barrier "
                 "forms in the range of a species.  The barrier prevents "
                 "genetic interchange and thus a previous one species can
turn"
                 " into two or more by genetic drift and different selective"
                 " pressures.",

                 "A classic case of Allopatric speciation is with chimpanzees"
                 " north of the Congo river and Bonobos south of it.",

                 "The Hox genes control body plan of an embryo from head to "
                 "tail in creatures as diverse from earthworms to fruit flies
"
                 "to humans.  That implies we all had a common ancestor "
                 "hundreds of millions of years ago.",

                 "The Krebs cycle is very important because it is how all "
                 "known aerobic organisms turn carbohydrates, fats and
protein"
                 " into energy."
              };


//  PURPOSE:  To make the professor wake up after receiving the
COMPLAIN_SIGNAL.
//  If the signal comes from the dean then sets 'awakeLevel = 2' and
//  prints:
```

```c
//          "Prof \"(Oops! The Dean caught me sleeping on the job!)\"\n"
//          "Prof \"Now as I was saying . . .\"\n"
//
//    If the signal *merely* comes from the student then sets 'awakeLevel =
1'
//    and prints:
//          "Prof \"Huh?  What?  Oops!  I must have fallen asleep!\"\n"
//          "Prof \"Now as I was saying . . .\"\n".
void        wakeUpHandler           (int             sigNum,
                                     siginfo_t*      infoPtr,
                                     void*           vPtr
                                    )
{
 // YOUR CODE HERE
}


//  PURPOSE:  To make this process stop by setting 'isStillClassTime = 0',
//    and to tell the student process to stop by sending it
//    CLASS_DISMISSED_SIGNAL.  Also prints:
//          "Prof \"Class dismissed!\"\n"
void        classDismissedHandler
                                    (int             sigNum
                                    )
{
 // YOUR CODE HERE
}


//  PURPOSE:  To install 'wakeUpHandler()' for 'COMPLAIN_SIGNAL' and
//    'classDismissedHandler()' for 'CLASS_DISMISSED_SIGNAL'.
void        installHandlers()
{
 // YOUR CODE HERE
}


//  PURPOSE:  To print the lesson, and to send 'PROF_TEACH_SIGNAL' to the
//    student.
void        teach                   (pid_t           studentPid,
                                     subject_ty      subject
                                    )
{
 const char**        cPtrPtr;

 switch  (subject)
 {
 case PHYSICS_SUBJECT :
  cPtrPtr    = PHYSICS_KNOWLEDGE;
  break;

 case CHEMISTRY_SUBJECT :
  cPtrPtr    = CHEMISTRY_KNOWLEDGE;
  break;

 case BIOLOGY_SUBJECT :
  cPtrPtr    = BIOLOGY_KNOWLEDGE;
```

```c
   break;
 }

 printf("Prof \"%s\"\n",cPtrPtr[rand() % NUM_FACTS_PER_SUBJECT]);
 // YOUR CODE HERE
}


//  PURPOSE:  To send the 'PROF_SNORE_SIGNAL' to the student.
void        snore           (pid_t          studentPid
                             )
{
 // YOUR CODE HERE
}


//  PURPOSE:  To start the student (child) process.  The child runs
//   'STUDENT_PROGRAM' with the command line argument 'text' (telling the
//    process id of the dean).  It prints
//           "Student \"I cannot find my classroom!\"\n"
//   and does:
//          exit(EXIT_FAILURE);
//   The parent process returns the process id of the student child
process.
pid_t         obtainStudent  ()
{
 char         text[MAX_LINE];
 pid_t        childPid;

 snprintf(text,MAX_LINE,"%d",getppid());

 // YOUR CODE HERE

 return(childPid);
}


//  PURPOSE:  To do the work of the professor.  Returns 'EXIT_SUCCESS'.
int         main           (int            argc,
                            char*          argv[]
                            )
{
 subject_ty        subject;
 struct sigaction  act;

 srand(getpid());
 installHandlers();

 if  ( (argc < 2)                                ||
       ( (subject = getSubjectFromName(argv[1])) == NO_SUBJECT)
     )
 {
  printf("Prof \"I don't know which course I'm teaching, I quit!\"\n");
  exit(EXIT_FAILURE);
 }

 studentPid = obtainStudent();
```

```
while  (isStillClassTime)
{
 if  (awakeLevel > 0)
 {
  teach(studentPid,subject);

  if  ( ((rand() % 1024) / 1024.0) < PROF_FALL_ASLEEP_PROB )
  {
   awakeLevel--;
  }

 }
 else
 {
  snore(studentPid);
 }

 sleep(1);
}

return(EXIT_SUCCESS);
      }
```

3. **Write `student.c`**

   `student.c` will have one command line argument: the process id of the dean. If this is
   missing then it should print:

   ```
   "Student \"Hey!  You did not tell me the Dean's number so I can complain
   if I need to\"\n"
   ```

   and then do `exit(EXIT_FAILURE);`

   The student should have 3 handlers:

   a. A `CLASS_DISMISSED_SIGNAL` handler. It ends the `sleep()` loop in `main()`. It
      should also display a message saying the student will quit:

      ```
      printf("Student \"Time for lunch!\"\n");
      ```

   b. A `PROF_TEACH_SIGNAL` handler. It just prints the following:

      ```
      printf("(Student scribbles notes.)\n");
      ```

   c. A `PROF_SNORE_SIGNAL` handler.

      This handler should keep track of two things:
```

1. The number of times the student was recently snored at
2. The number of times the student recently complained to the prof

Every time it is called it should increment the number of times the student was recently snored at.

1. If this is less than `NUM_SNORES_BEFORE_STUDENT_COMPLAINS` then it should just print:

```
printf("Student \"%s?\"\n",PROFS_NAME);
```

2. Else, it should
   a. Reset the number of times the student was recently snored at to `0`
   b. Increment the number of times the student recently complained to the prof.
   c. If that count is less than `NUM_COMPLAINTS_TO_PROF_BEFORE_COMPLAIN_TO_DEAN` then it should send `COMPLAIN_SIGNAL` to the prof process (its parent) and print

```
printf("Student \"%s!\"\n",PROFS_NAME);
```

   d. If the count is greater than or equal to `NUM_COMPLAINTS_TO_PROF_BEFORE_COMPLAIN_TO_DEAN` then it should send `COMPLAIN_SIGNAL` to the dean process, set the count to `0` and print

```
printf("Student \"I have had enough!  I'm calling the Dean.\"\n");
```

After installing the handlers, `main()` should just do a `sleep()` loop waiting for the program to end. I had:

```
while  (isStillClassTime)
{
 sleep(1);
}

return(EXIT_SUCCESS);
```

How to make the program:

```
$ gcc -c subjectName.c
$ gcc -c dean.c
```

```
$ gcc -c prof.c
$ gcc -o dean dean.o subjectName.o
$ gcc -o prof prof.o subjectName.o
$ gcc -c student.c
$ gcc -o student student.o
```

Then run it with the following:

```
$ ./dean
```

The `dean` starts the `prof`. The `prof` starts the `student`.

Sample output:

```
$ ./dean
Dean "Welcome to the Bubu University!"
Dean "What would you like to study?"
 1: Physics
 2: Chemistry
 3: Biology
1
Dean: "Prof Narcolepsy, today you will teach Physics"
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
Prof "(Oops! The Dean caught me sleeping on the job!)"
Prof "Now as I was saying . . ."
(The Dean shuffles paper.)
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
Prof "Fermions have 1/2 spin and cannot occupy the same quantum state, Bosons
ha
ve integer spin and can occupy the same quantum state."
(Student scribbles notes.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
(The Dean shuffles paper.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
```

on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
Prof "Fermions have 1/2 spin and cannot occupy the same quantum state, Bosons
ha
ve integer spin and can occupy the same quantum state."
(Student scribbles notes.)
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
(The Dean shuffles paper.)
Prof ("Snore!")
Student "Prof Narcolepsy?" **(1st time snoring, no signal yet)**
Prof ("Snore!")
Student "Prof Narcolepsy?" **(2nd time snoring, no signal yet)**
Prof ("Snore!")
Student "Prof Narcolepsy!" **(3rd time snoring, now we send signal)**
          (1st time we had to wake the prof)
Prof "Huh? What? Oops! I must have fallen asleep!"
Prof "Now as I was saying . . ."
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
(The Dean shuffles paper.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)

```
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
Prof ("Snore!")
Student "Prof Narcolepsy?"
(The Dean shuffles paper.)
Prof ("Snore!")
Student "Prof Narcolepsy?"
Prof ("Snore!")
Student "Prof Narcolepsy!" (2nd time we had to wake the prof)
Prof "Huh?  What?  Oops!  I must have fallen asleep!"
Prof "Now as I was saying . . ."
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
(The Dean shuffles paper.)
Prof ("Snore!")
Student "Prof Narcolepsy?"
Prof ("Snore!")
Student "Prof Narcolepsy?"
Prof ("Snore!")
Student "I have had enough!  I'm calling the Dean."
        (3rd time, we give up of the prof and call the dean)
Dean "WAKE UP!"
Prof "(Oops! The Dean caught me sleeping on the job!)"
Prof "Now as I was saying . . ."
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
(The Dean shuffles paper.)
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
```

on, then electrons collapse into the nucleus and matter becomes a neutron
star."
(Student scribbles notes.)
Prof "The electron-electron repulsion between atoms and molecules supports the
s
tructure of most matter at our scale. If gravity overcomes this electron
repulsi
on, then electrons collapse into the nucleus and matter becomes a neutron
star."
...
Prof "There is a large Black hole at the center of our galaxy."
(Student scribbles notes.)
Prof "F = m*a is a special case of F = dp/dt = m*dv/dt + v*dm/dt when dm/dt =
0.
"
(Student scribbles notes.)
Dean "Class is over!"
Prof "Class dismissed!"
Student "Time for lunch!"
(The Dean turns of the lights and locks the classroom.)