ECS 30 **Homework #1 (31 points)** **Fall 2013**

Due: Wednesday, October 9th. Written: 4pm in 2131 Kemper. Programs: 11:55pm using handin to cs30, p1 directory.
Filenames: reimbursement.c, score.c, grass.c, triple.c, jet.c
Written (6 points)

1. based on p. 101, #2 Which variables below are syntactically correct?
   `income, 2time, int, Tom's, two fold, c3po, income#1, item`

2. p. 102, #4 Stylistically, which of the following identifiers would be good choices for names of constant macros?
   `gravity  G  MAX_SPEED  Sphere_Size`

3. (3 points) p. 102, #5 Write the data requirements, necessary formulas, and algorithm for Programming Project 9 (see grass.c below). This does NOT mean write the program. You should make three lists with the aforementioned headings.

4. p. 102, #7 List three standard data types of C.

Programming (25 points) from Chapter 2, pp. 102-106.

All programs should be able to compile with no warnings when compiled with the –Wall option, e.g. gcc –Wall reimbursement.c. You should put your name(s) in a comment on the first line of each file. In the examples, the user input is in **bold.**

The prompts, and output format of each program must match my programs **EXACTLY**. You will find my executables and some testing files in ~ssdavis/30/p1 in the CSIF. Note that these are not accessible from the web. You must log into your CSIF account and use cp to copy them to your own directory, e.g. `cp ~ssdavis/30/p1/*  p1` where you have already created a p1 directory.

You can use the UNIX `diff` utility to compare your executable with mine. `diff` will compare two files line by line, and list the changes needed that need to be made to the first file to make it identical to the second file. If there are no differences between the two files then `diff` outputs nothing.

In the following example of using `diff` I altered my reimbursement.c source code to produce an incorrect prompt, an incorrectly formatted number, and an incorrect value, and named the executable reimbursementErrors.out. To ensure that both executables receive the same input, I typed the input into a file named reimbursement1.txt, and then redirected it into each executable using the "<." To redirect the output from the monitor into files, I used the ">."

```
[ssdavis@lect1 p1]$ cat reimbursement1.txt
13505.2
13810.6

[ssdavis@lect1 p1]$ reimbursement.out < reimbursement1.txt > seans.txt
[ssdavis@lect1 p1]$ reimbursementErrors.out < reimbursement1.txt > errors.txt
[ssdavis@lect1 p1]$ cat seans.txt
MILEAGE REIMBURSEMENT CALCULATOR
Enter beginning odometer reading=> Enter ending odometer reading=> You traveled 305.4 miles. At $.35 per mile,
your reimbursement is $106.89.
[ssdavis@lect1 p1]$ cat errors.txt
MILEAGE REIMBURSEMENT CALCULATOR
Enter beginning odometer reading => Enter ending odometer reading=> You traveled 305.40 miles. At $.35 per mile,
your reimbursement is $113.00.
[ssdavis@lect1 p1]$ diff errors.txt seans.txt
2,3c2,3
< Enter beginning odometer reading => Enter ending odometer reading=> You traveled 305.40 miles. At $.35 per mile,
< your reimbursement is $113.00.
---
> Enter beginning odometer reading=> Enter ending odometer reading=> You traveled 305.4 miles. At $.35 per mile,
> your reimbursement is $106.89.
[ssdavis@lect1 p1]$
 [ssdavis@lect1 p1]$ diff seans.txt seans.txt
[ssdavis@lect1 p1]$
```

#1. Filename: reimbursement.c

"Write a program that calculates mileage reimbursement for a salesperson at a rate of $.35 per mile. Your program should interact in this manner:

```
MILEAGE REIMBURSEMENT CALCULATOR
Enter beginning odometer reading=> 13505.2
Enter ending odometer reading=> 13810.6
You traveled 305.4 miles. At $.35 per mile,
your reimbursement is $106.89."
```

You should use a macro to store the $.35 rate in a constant. However, there is no simple way to print the rate as "$.35", instead of "$0.35", so do not try to use "%lf" with the constant—just printf "$.35" literally.

```
[ssdavis@lect2 p1]$ reimbursement.out
MILEAGE REIMBURSEMENT CALCULATOR
Enter beginning odometer reading=> 13505.2
Enter ending odometer reading=> 13810.6
You traveled 305.4 miles. At $.35 per mile,
your reimbursement is $106.89.
[ssdavis@lect2 p1]$
[ssdavis@lect2 p1]$ reimbursement.out
MILEAGE REIMBURSEMENT CALCULATOR
Enter beginning odometer reading=> 235892.45
Enter ending odometer reading=> 248111.329
You traveled 12218.9 miles. At $.35 per mile,
your reimbursement is $4276.61.
[ssdavis@lect2 p1]$
```

#6 Filename: score.c

"Write a program that predicts the score needed on a final exam to achieve a desired grade in a course. The program should interact with the user as follows:

```
Enter desired grade> B
Enter minimum average required> 79.5
Enter current average in course> 74.6
Enter how much the final counts
as a percentage of the course grade> 25

You need a score of 94.20 on the final to get a B."
```

```
[ssdavis@lect2 p1]$ score.out
Enter desired grade> B
Enter minimum average required> 79.5
Enter current average in course> 74.6
Enter how much the final counts
as a percentage of the course grade> 25

You need a score of 94.20 on the final to get a B.
[ssdavis@lect2 p1]$
```

```
[ssdavis@lect2 p1]$ score.out
Enter desired grade> A
Enter minimum average required> 90.0
Enter current average in course> 86.3
Enter how much the final counts
as a percentage of the course grade> 30

You need a score of 98.63 on the final to get a A.
[ssdavis@lect2 p1]$
```

#9. Filename: grass.c "Write a program that takes the length and width of a rectangular yard and the length and width of a rectangular house situated in the yard. Your program should compute the time required to cut the grass at the rate of two square feet a second." The numbers are integers.

```
[ssdavis@lect2 p1]$ grass.out
Please enter the length and width of the yard > 100 95
Please enter the length and width of the house > 45 38
7790 square feet will take 64 minutes and 55 seconds to cut.
[ssdavis@lect2 p1]$
[ssdavis@lect2 p1]$ grass.out
Please enter the length and width of the yard > 40 70
Please enter the length and width of the house > 38 62
444 square feet will take 3 minutes and 42 seconds to cut.
[ssdavis@lect2 p1]$
```

#11. Filename: triple.c "The Pythagorean theorem states that the sum of the squares of the sides of a right triangle is equal to the square of the hypotenuse. For example, if two sides of a right triangle have lengths of 3 and 4, then the hypotenuse must have a length of 5. Together the integers 3, 4, and 5 form a *Pythagorean triple.* There are an infinite number of such triples. Give two positive integers, *m* and *n*, where $m > n$, a Pythagorean triple can be generated by the following formulas:

$side1 = m^2 - n^2$
$side2 = 2mn$
$hypotenuse = m^2 + n^2$

The triple (*side1 = 3, side2 = 4, hypotenuse* =5) is generated by this formula when $m = 2$ and $n = 1$. Write a program that takes values for *m* and *n* as input and displays the values of the Pythagorean triple generated by the formulas above."

```
[ssdavis@lect2 p1]$ triple.out
Please enter two integers with the first larger than the second > 2 1
The Pythagorean triple of 2 and 1 is 3, 4, and 5.
[ssdavis@lect2 p1]$
[ssdavis@lect2 p1]$ triple.out
Please enter two integers with the first larger than the second > 15 9
The Pythagorean triple of 15 and 9 is 144, 270, and 306.
[ssdavis@lect2 p1]$
```

#12. Filename: jet.c "Write a program that calculates the acceleration (m/s^2) of a jet fighter launched from an aircraft-carrier catapult, given the jet's takeoff speed in km/hr and the distance (meters) over which the catapult accelerates the jet from rest to takeoff. Assume constant acceleration. Also calculate the time (seconds) for the fighter to be accelerated to takeoff speed. When you prompt the user, be sure to indicate the units for each input. For one run, use a takeoff speed of 278 km/hr and a distance of 94 meters. Relevant formulas (*v* = velocity, *a* = acceleration, *t* = time, *s* = distance)

$v = at$

$s = \frac{1}{2}at^2$

```
[ssdavis@lect2 p1]$ jet.out
Please enter the takeoff speed of the jet in km/hr > 278
Please enter the catapult distance in meters > 94
The jet would accelerate at 31.72 m/s^2 for 2.43 seconds.
[ssdavis@lect2 p1]$
[ssdavis@lect2 p1]$ jet.out
Please enter the takeoff speed of the jet in km/hr > 137
Please enter the catapult distance in meters > 115
The jet would accelerate at 6.30 m/s^2 for 6.04 seconds.
[ssdavis@lect2 p1]$
```

Submitting files
Usage: handin touser subdirectory files...

For our class the touser is cs30. The subdirectories are tutorial, p1, p2, p3, p4, p5, p6, p7, and p8. The files are the names of the files you wish to submit. For this assignment, your files will be reimbursement.c, score.c, grass.c, triple.c, and jet.c. Therefore, you would type:

handin cs30 p1 reimbursement.c, score.c, grass.c, triple.c, jet.c

or

handin cs30 p1 *.c

This will copy all the specified files to ~cs30/handin/p1/from_user, where from_user is the username of the person doing the submitting. The directory from_user is made if it does not already exist and is named after the invoking user. Therefore, each user can submit files with the same names as other users and there will be no overwriting, because each user has his/her own subdirectory. If you resubmit a file with the same name as one you had previously submitted, the new file will overwrite the old file. This is handy when you suddenly discover a bug after you have used handin. But beware, the time stamps of the files in the handin directories are those of the time of submission, and not those of the time of file creation. A file you wrote yesterday, but handin today, will have today's date and submission time. DO NOT use handin after the due date time. Files with late times will NOT be graded.