CS 1713, Assignment 1, Spring 1998, due January 20
Copy Program, Quadratic Equations

The assignment: Copy the program, compile and run it, and turn in a listing and results of the run.

```
/* Name:       Insert your name here
 * Date:       Insert the due date here
 * Course:     CS 1713 Section n
 * Subject:    Calculate the roots of a quadratic equation.
 * Algorithm:  Read coefficients from the terminal. Use the quadratic formula.
 *             Only real roots are calculated.
 * Input:      Enter three coefficients in double format.
 * Output:     If the roots are real, display them.
 *             Otherwise display an error message.
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void main(int argc, char *argv[])
{
    double a0, a1, a2;  /* coefficients in a2 x^2 + a1 x + a0 = 0 */
    double discriminant;  /* a1*a1 - 4*a2*a0 */
    double root1, root2;  /* the two roots of the equation */
    /* prompt for the coefficients, if not on command line */
    if (argc == 1) {
        fprintf(stdout, "Solving a2 x^2 + a1 x + a0 = 0\n");
        fprintf(stdout, "Enter a2, a1, a0 (in order): ");
        scanf("%lf %lf %lf", &a2, &a1, &a0);
    }
    else if (argc == 4) {  /* get coefficients from command line */
        a2 = atof(argv[1]);
        a1 = atof(argv[2]);
        a0 = atof(argv[3]);
    }
    else {
        fprintf(stderr, "%s requires 0 or 3 arguments\n", argv[0]);
        exit(1);
    }
    fprintf(stdout, "Solving %5.3f x^2 + %5.3f x + %5.3f = 0\n",
          a2, a1, a0);
    if (a2 == 0.0) {  /* this is a linear equation (no quadratic term) */
        fprintf(stderr, "a2 must not be 0 in a quadratic equation\n");
        exit(1);
    }
    discriminant = a1*a1 - 4.0*a0*a2;
    if (discriminant < 0.0) {
        fprintf(stderr, "Roots are complex numbers\n");
        exit(1);
    }
    /* calculate roots and output them */
    root1 = (-a1 + sqrt(discriminant))/(2.0*a2);
    root2 = (-a1 - sqrt(discriminant))/(2.0*a2);
    fprintf(stdout, "Roots: %5.3f, %5.3f\n", root1, root2);
    exit(0);
}
```

What to do: (In the text that follows, **boldface** is what you should enter at the keyboard.)

1. Connect to the machine: look for a terminal with the "`local>`" prompt, or connect to UTSA by modem. You may have to type RETURN to get the prompt: `Local>` **`c runner`**
   You should now get the prompt: `login:`
2. Carry out the login to your account: type in your account name: `login:` **`YourAccountName`**
   You should get the prompt: `Password:`
   Next enter your password, initiallyyour 9-digit SSN without hyphens: `Password:` **`YourSSN`**
3. You are now on the system. The first time you will be asked to choose a new password. The new password must be 6 to 8 characters long, and must contain at least 2 alphabetic characters and at least 1 numeric or special character. The system will prompt you for your old password (your SSN) and your new password (twice, to check that you have it correct). You will see the following on the screen:

```
Your password has expired.  Choose a new one.
Old password: YourSSN
```

```
New password: YourNewPassword
Re-enter new password: YourNewPassword
```

(Blanks and upper or lower-case letters count in these passwords and elsewhere. *Do not forget* your new password.) You are now on the system and should get the system prompt: runner%

4. Now create a file called `roots.c` which will contain the copy program for the first assignment, listed above. (This program reads in 3 numbers that are used as the coefficients of a quadratic equation. The program then calculates and prints the roots of this equation.) Read about the *vi* editor in the UNIX Primer book. To use *vi*, type: runner% **`vi roots.c`**
5. When you have finished typing the program, you should first check it for correctness with the `lint` utility: `runner%` **`lint -m -u roots.c -lm`**
   (Don't leave off the "magic" hyphen-el-em at the end.)
   If you have typed in the program correctly, you should get the following messages from `lint`, which you can ignore for now:
   ```
   function returns value which is always ignored
         fprintf     scanf
   ```
   You must carefully type exactly the characters given for the program. If there are any typing mistakes, you will probably get error messages from lint.
6. Next you should compile the program `roots.c` to produce an executable module called `roots`. For this type the command: runner% **`cc -o roots roots.c -lm`**
   There are a number of "magic" parts to this command which must be just right. The "`cc`" invokes the C compiler. The "`-o roots`" says to name the executable file "`roots`". Finally the same "`-lm`" as with lint above tells the system where to find the `sqrt` function in `roots.c`, that calculates the square root. Everything must be exactly right, so that it must be a lower-case "Oh" in "`-o`" and it must be a lower-case "El" in "`-lm`", and not a zero or a one. There should be no messages in response to this command. If you have not typed in the program correctly, then as with lint, you will probably get error messages.
7. Now you can try executing your program, in one of two ways, either with the three numbers on the same line that starts up the program, or with the three numbers typed later. In Unix we execute a file by just typing the name of the file. After execution completes you will get the "runner%" prompt again.
   ```
   runner% roots 1.0 -2.0 -3.0
   Solving 1.000 x^2 + -2.000 x + -3.000 = 0
   Roots: 3.000, -1.000
   runner% roots
   Solving a2 x^2 + a1 x + a0 = 0
   Enter a2, a1, a0 (in order): 1.0 -2.0 -3.0
   Solving 1.000 x^2 + -2.000 x + -3.000 = 0
   Roots: 3.000, -1.000
   runner%
   ```
   You should try entries that result in complex roots, and an entry with first coefficient zero.
8. When you have gotten this far, you can produce a listing to hand in. For this purpose, you should use the `script` command, which inserts every character that appears on the screen into a file named "`typescript`". Script is terminated with `Control-d` (hold the Control key down and type "`d`"). Inside script you should type "`cat roots.c`" to cause your program text to appear on the screen. Then you should execute the program with inputs as above. A sample session might look like:
   ```
   runner% script
   Script started, file is typescript
   runner% cat roots.c
   /* Name:       Insert your name here
   * Date:       Insert the due date here
       ... (lines omitted)
         exit(0);
   }
   runner% roots 1.0 -2.0 -3.0
   Solving 1.000 x^2 + -2.000 x + -3.000 = 0
   Roots: 3.000, -1.000
   runner% roots
   Solving a2 x^2 + a1 x + a0 = 0
   Enter a2, a1, a0 (in order): 1.0 -2.0 -3.0
   Solving 1.000 x^2 + -2.000 x + -3.000 = 0
   Roots: 3.000, -1.000
   runner% Control-d Script done, file is typescript
   ```
9. Now finally you can get a printout to hand in for this assignment by typing:
   ```
   runner% lp -d SCF1 typescript
   ```
   Put your name at the front of this listing and hand it in on the due date.
10. When you are done you should logout by typing exit: runner% **`exit`**