CS270 Recitation 3
"C Debugging Exercise"

Goals

To learn debugging a C program using the GNU debugger tool: `gdb`. You will use both the graphical front-end, as well as the command line to debug the sample program.

The Assignment

Make a subdirectory called R3 for the recitation inside your cs270 directory. All files should reside in this subdirectory.

```
%> mkdir R3
```

Copy the file `http://www.cs.colostate.edu/~cs270/Recitations/R3/r3.c` into your R3 subdirectory (we will use a command line HTTP client for this purpose).

```
%> wget http://www.cs.colostate.edu/~cs270/Recitations/R3/r3.c
```

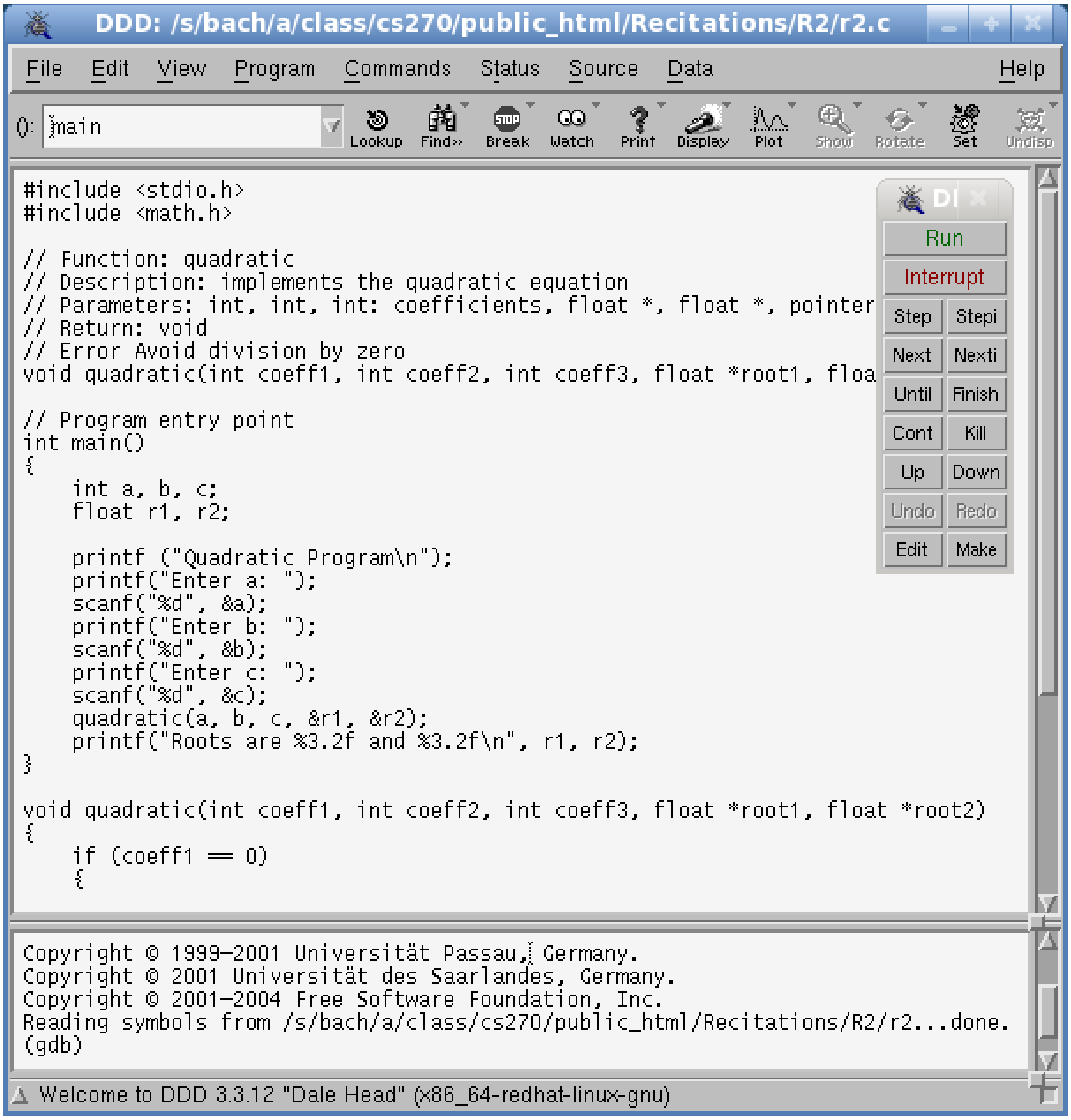
Compile the program into an executable called r3, as shown below.

```
%> gcc -g -std=c99 -lm -Wall r3.c -o r3
```

To debug the compiled program, type the following command:

```
%> ddd r3
```

This command opens up the graphical front end for gdb (see the screenshot below). You can type gdb commands at the prompt inside DDD.



```
#include <stdio.h>
#include <math.h>

// Function: quadratic
// Description: implements the quadratic equation
// Parameters: int, int, int: coefficients, float *, float *, pointer
// Return: void
// Error Avoid division by zero
void quadratic(int coeff1, int coeff2, int coeff3, float *root1, floa

// Program entry point
int main()
{
    int a, b, c;
    float r1, r2;

    printf ("Quadratic Program\n");
    printf("Enter a: ");
    scanf("%d", &a);
    printf("Enter b: ");
    scanf("%d", &b);
    printf("Enter c: ");
    scanf("%d", &c);
    quadratic(a, b, c, &r1, &r2);
    printf("Roots are %3.2f and %3.2f\n", r1, r2);
}

void quadratic(int coeff1, int coeff2, int coeff3, float *root1, float *root2)
{
    if (coeff1 == 0)
    {
```

```
Copyright © 1999-2001 Universität Passau, Germany.
Copyright © 2001 Universität des Saarlandes, Germany.
Copyright © 2001-2004 Free Software Foundation, Inc.
Reading symbols from /s/bach/a/class/cs270/public_html/Recitations/R2/r2...done.
(gdb)
```

Welcome to DDD 3.3.12 "Dale Head" (x86_64-redhat-linux-gnu)

Use the following debugger commands to run the program and examine variables:

```
(gdb) set logging on      // enable logging to gdb.txt
(gdb) break 26            // set breakpoint at line 26
(gdb) break 27            // set breakpoint at line 27
(gdb) break 28            // set breakpoint at line 28
(gdb) run                 // run program
```

Enter integer values for a, b, and c as requested by the program.

```
(gdb) print a             // print value of a
(gdb) print b             // print value of b
(gdb) print c             // print value of c
(gdb) print r1            // print value of r1
(gdb) print r2            // print value of r2
```

```
(gdb) step                // step one line
(gdb) break 47            // set breakpoint at line 47
(gdb) continue            // continue to breakpoint just set
(gdb) print rootValue     // print value of rootValue
(gdb) info breakpoints    // list all breakpoints set above
(gdb) disable 1           // disable first breakpoint
(gdb) info breakpoints    // list all breakpoints set above
(gdb) enable 1            // reenable first breakpoint
(gdb) delete 1            // delete first breakpoint
(gdb) info breakpoints    // list all breakpoints set above
(gdb) help                // show help message command categories
(gdb) help status         // show help message for status commands
(gdb) info stack          // display stack status
(gdb) info functions      // display function status
(gdb) continue            // continue to next breakpoint
(gdb) info locals         // display local variables status
(gdb) quit                // quit debugger
```

Display the output of your debugging session and show it to the TA:

```
%> less gdb.txt
```