CIS 403: Problem Set 2

Directions for Submission

E-mail your answers to me at ajp9@cornell.edu. The subject of your message should be “CIS403 PS2,” and the body of the message should contain your answers.

Essential Knowledge–Please give a brief answer (1-2 sentences) for each

1. Describe a subtle bug that a C compiler would catch, but a FORTRAN compiler would miss.
   Because it has prototypes, a C compiler can catch some incorrect subroutine calls. FORTRAN doesn’t even try.

2. Your lab comparing FORTRAN and C was designed to paint FORTRAN in an unfavorable light. Describe some advantages of poor, old FORTRAN.
   In my opinion, FORTRAN’s main advantage (and main reason for its continued existence) is the volume of legacy code still available. This is especially true for libraries for numerical computing.

3. Here are a couple of lines from a Makefile:
   
   f1.o:f1.c f1.h f2.h
   $(CC) $(CFLAGS) -w f1.c

   Describe what would happen if you change f1.c and type make? What if you change f3.h?
   Changing f1.c will cause f1.o to be built according to the statement on the second line. Since f3.h does not appear in the dependency statement, changing f3.h will not cause f1.o to be built.

4. You’ve been given a program with the following statement:

   
   ```
   :;
   val[j]=input[right]-input[left];
   :;
   ```
where \texttt{val} and \texttt{input} are both length \( m \) arrays (\( m > 0 \)). The code compiles, but when you run the program, it crashes. The code fragment is embedded in a complicated \texttt{while} loop that does a lot of funny things to the value of \( j \), \texttt{left}, and \texttt{right}. You suspect that there is a bug that causes the value of \texttt{left} and/or \texttt{right} to be \(< 0 \) or \( \geq m \). Describe how you would test your hypothesis using either \texttt{gdb} or checkpointing.

Setting a breakpoint at this line and then printing \texttt{right} and \texttt{left} would work, but would be tedious if \( m \) is very large. Printing out the value of \texttt{right} and \texttt{left} before this line would also work, but could be equally tedious. Comparing the value of \texttt{left} and \texttt{right} with \( m \) and 0 and printing out all relevant variables if there is an error would be my preference:

\begin{verbatim}
if(left <0 || left >=m || right <0 || right >=m){
    printf("Problem with left and right:\n");
    /* Print left, right, m, j and any 
other interesting variables */
    :
}
val[j]=input[right]-input[left];
:
\end{verbatim}