1 Overview

This is the same database problem described in the lab 6; however, you will modify your makefile and program to optimize their functionality and performance. First, you will create a makefile that is more sophisticated and versatile. Afterwards, you will evaluate the performance of your program and make modifications to enhance speed and efficiency. You must adhere to the lab 6 program description (dynamically size the arrays when elements are added or deleted).

2 Makefiles

As you know, make is a utility that performs a series of commands described in a special formatted file. The file makefile (or Makefile) is the default make uses; however, the make option -f allows you to specify any filename. A makefile contains entries of the form

```
  targetName: dependencyList
  commandLine
```

The `targetName` (what the command should generate) is followed by a colon, which is followed by a `dependencyList` (what must be newer than the target to execute the command). The next line consists of a `commandLine`, that should produce the `targetName` (remember, command lines must begin with a tab).

2.1 makefile Comments

Placing the character # at the beginning of a line in the makefile creates a comment. The make utility will skip these lines and proceed to the first dependency rule. For the remaining lab assignments this semester, you must comment your makefiles. Provide your name, the course, the assignment number, and date. For example consider the following makefile

```
# Programmer: Nomad Noedia Course: CSC112 B
# Assignment: 7 Date: 10/10/2002

driver: driver.o stats.o
g++ -o driver driver.o stats.o
driver.o: driver.cpp stats.h
g++ -c driver.cpp
stats.o: stats.cpp stats.h
g++ -c stats.cpp
```

2.2 make clean

Often you need to erase all the object files associated with an assignment and compile with a clean slate. This can be done using a special target called clean that is placed at the end of the makefile.

```
# Programmer: Nomad Noedia Course: CSC112 B
# Assignment: 7 Date: 10/10/2002

driver: driver.o stats.o
g++ -o driver driver.o stats.o
driver.o: driver.cpp stats.h
g++ -c driver.cpp
stats.o: stats.cpp stats.h
g++ -c stats.cpp
clean:
  rm -f *.o driver
```
To execute this option enter the command \texttt{make clean} at the prompt. In the preceding example, all the .o files and the executable driver will be erased. Do not erase source files! Note there is no dependency list, so the commands that follow will always be executed. For the remaining lab assignments this semester, your \texttt{makefile} must include a \texttt{clean} option.

2.3 \texttt{makefile} Variables

Often a \texttt{makefile} consists of the same options and commands. We can represent these items using \texttt{make} variables. Variable (macro) definitions begin at the left margin of the \texttt{makefile} and have the form

\texttt{variableName = variableDefinition}

For example, our \texttt{makefiles} will consistently use the \texttt{g++} command for compiling. We could use the following \texttt{make} variable to represent this item

\texttt{CC = g++}

Now the variable \texttt{CC} represents the command \texttt{g++}. Anytime we need to use the \texttt{g++} command in the \texttt{makefile}, we would use \texttt{$(CC)} instead. \textit{This does not appear to shorten anything, so why should I use it?} Consider the case where you need to compile your program with another compiler. In this situation, you only have to change the \texttt{CC} definition. Another use for \texttt{make} variables is compiler flags. For example, if we always compile using \texttt{-g}, \texttt{-ansi}, and \texttt{-pedantic} we could use a single variable,

\texttt{CFLAGS = -g -ansi -pedantic}

The \texttt{-ansi} and \texttt{-pedantic} compiler flags ensure your code meets the C++ ANSI standards. So compare our new and improved \texttt{makefile} to the old style.

The new and improved example:

\begin{verbatim}
# Programmer: Nomad Nocnaed Course: CSC112 B
# Assignment: 7 Date: 10/10/2002

CC = g++
CFLAGS = -g -ansi -pedantic
driver: driver.o stats.o
   $(CC) -o driver driver.o stats.o
driver.o: driver.cpp stats.h
   $(CC) $(CFLAGS) -c driver.cpp
stats.o: stats.cpp stats.h
   $(CC) $(CFLAGS) -c stats.cpp
clean:
   \rm -f *.o driver
\end{verbatim}

The old and inflexible example:

\begin{verbatim}
# Programmer: Anil Ratan Leebrat Course: CSC112
# Assignment: 7 Date: 10/10/2002

driver: driver.o stats.o
   g++ -o driver driver.o stats.o
driver.o: driver.cpp stats.h
   g++ -g -ansi -pedantic -c driver.cpp
stats.o: stats.cpp compl.h
   g++ -g -ansi -pedantic -c stats.cpp
clean:
   \rm -f *.o driver
\end{verbatim}

2.4 What to Turn-in

Go to your Lab6 directory and update your \texttt{makefile}. Your updated version must include the following:

- Comments indicating your name, course, assignment, and date
- \texttt{clean} target that removes the object files and the executable
- Variables for the compile command (\texttt{g++}) and the \texttt{-g} compiler flag. Do not add the \texttt{-ansi} or \texttt{-pedantic} flags for this assignment.

Therefore, your updated \texttt{makefile} will be similar to the new and improved example above.