

Using Process Journals to Gain Qualitative Understanding of Beginning Programmers

Gary Lewandowski

Department of Mathematics and Computer Science

Xavier University

lewandow@cs.xu.edu

Abstract

Process journals are informal journals kept by students as they work on a programming project. The goal in using them is to gain qualitative information about the student's problem solving and programming process. This paper presents a qualitative analysis of the journals from a data structures course, categorizing the broad themes found in the journals and the nature of the responses given by the instructor. The goal of this qualitative analysis is to answer the questions, "what sort of information do you see in a process journal and how do you use them in your course?"

1 Introduction

Performance on a programming project assesses a student's ability to complete a substantial concrete task. It does not assess the reasons for success or failure. Did the student not spend enough time? Or did (s)he spend a great deal of time in fruitless frustration? Is the student an excellent problem solver? Or is the student simply good at networking with others in the course? Is the student's workstyle appropriate for long-term success? Are there obvious suggestions we can give that will help this student get better? The answers to these questions require qualitative data about the student's approach to the project.

Concerns about retention overall and student problem-solving skills in upper level courses lead us to ask many of the questions above. As we modified our early programming courses

to use more open-ended projects, we needed to know what students were doing as they succeeded or failed to complete them. We began assigning *process journals* as a part of each programming project in the early programming courses. Process journals are student logs tracking time, activity and thoughts as a student works on a programming project. To encourage an honest and informative account the journal's structure is very informal beyond rudimentary time logging. In each project, the journal is listed as a deliverable, with the following description:

“A programming journal, kept electronically. Every time you work on the project add an entry indicating

- The time you started.
- The time you stopped.
- What you did, e.g. design, debugging, testing, etc. Give details! It is especially useful to note when you had to redo things, or what problems were particularly hard. The Journal is a good place to keep your rough designs as you work them out. The point of a journal is to help both me and you understand your thinking process. This helps me give you better feedback. They also help you get going faster each day by giving a summary of where you left off. Sometimes they're cathartic as well.”

We have used the journals for three years and believe they help improve instruction and feedback to students. When we discuss these journals with colleagues we are commonly asked about the type of information we see and how we use that information. Our answers have been intuitive and anecdotal in these informal conversations. In this paper, we provide an answer to these questions based on a more careful qualitative analysis of a subset of the journals. We hope the answers will allow others to use journals of this sort productively, will suggest modifications to their use, and suggest new questions that can be studied using the data in the process journals.

Below we introduce our method of using process journals, distinguish them from other types of journaling efforts, particularly the personal software process (PSP) methodology [5], and reflective diaries [3], and then present our qualitative analysis of the journals and their use.

2 Background

Other approaches to capturing the student process have been used. The most significant is the Personal Software Process [5] which uses time-management as a driving force to have students track their process. While providing much of the information we are interested in (time spent, activities, design/coding bugs) we were not sure PSP would provide direct insight into strategies the students were actually using, collaborations they were forming, or confidence issues. PSP also involves some discipline and training time; we decided to try a less formal approach.

Fekete et al. [3] use reflection in their curriculum as well, using weekly reflective diaries that include descriptions of what students have done and their confidence in their understanding of concepts. Process journals are intended to be more immediate, capturing the student's process and feeling as they work.

Many researchers have conducted formal studies of the novice programming process. Winslow [6] gives an overview of what is known. The studies themselves often involve direct observation on particular tasks. For example, Gugerty and Olson [4] gave experts and novices particular debugging problems and studied how quickly, if at all, the participants found the bug, and captured the screen and keystrokes to determine what strategies were being used. Davis et al. [2] gave students LISP tasks and had them also speak aloud about their process as they did the problems, then used this information to analyze the student process according to rules they might be using to solve the problems. An advantage of these studies is that an expert is a controlled study and in the Davis et al. case we can be sure the student is giving an immediate response to the process. The disadvantage to these studies is that we would like similar information about *every* student but the interview process is too expensive. Since the tasks given in the studies do not impact one's grade and are usually short-term, we do not see how the student progresses over time on the project or how interaction with others affects their progress. Process journals give us a window into each student's process; combined with information from the literature the journals can provide an understanding of appropriate feedback or intervention for a given student.

3 Methodology

Process journals are used in our two introductory programming courses. In our curriculum these are the second and third course taken by the students after a breadth-first overview of the field that includes a small amount of programming along with an introduction to algorithmic thinking, circuits, von Neumann machines and readings relating technology to society.

Along with the description given above, students are shown one example journal from the first semester the journals were used [1]. The example journal is very conversational and details the process quite thoroughly. While it may serve as a biasing agent in terms of the information received, we believe it still yields natural observations from students because other students during the first semester produced journals of a similar style and because the conversational tone and content suggest a brutally honest discussion from the writer.

Students were assigned the journal as a part of each programming project in their course. As many as five points (out of 100 for the assignment) were awarded for the journals. Students received fewer points if it did not appear they were putting even the minimal information of time they worked in the journal or if it seemed like they were writing the journal after the fact.

To characterize the information provided by students we read the journals of twenty-one students in the second programming course (Data Structures) for five projects each. Some journals were turned in on paper and were not read for this study; some students dropped the course or did not turn in logs. Altogether, eighty-nine journals were read, sixty by males and twenty-nine by females. After an initial reading to pull out broad themes, the journals were re-read to provide a categorization of the journal as having the theme or not. In addition to this analysis of the information recorded in the journals, we read the responses to the students by the instructor and pulled out the main uses of the journals in the responses.

4 Analysis

4.1 Qualitative Analysis

The fifteen most frequent themes emerging in the journals are described in table one. The table gives the number of journals fitting the category, divided into male and female re-

sponses.

Theme	Journals exhibiting theme (%)	
	males	females
describe tasks completed	44 (0.73)	22 (0.76)
describe their approach to solving problems	41 (0.68)	17 (0.59)
describe in detail a bug or problem	36 (0.6)	16 (0.55)
indicate emotional responses to the process	28 (0.47)	15 (0.52)
mention discussion with peers or getting help	26 (0.43)	17 (0.59)
explain how they started	27 (0.45)	14 (0.48)
note significant roadblocks in design, coding or debugging	26 (0.43)	14 (0.48)
discuss design of the code	20 (0.33)	17 (0.59)
mention without detail a bug or problem	22 (0.37)	13 (0.45)
note progress tracking bug or problem	21 (0.35)	13 (0.45)
note taking breaks	19 (0.32)	9 (0.31)
discuss their use of time	23 (0.38)	5 (0.17)
mention testing	12 (0.2)	5 (0.17)
discuss rewriting code	8 (0.13)	5 (0.17)
make confidence or intelligence-related statements	9 (0.15)	3 (0.10)

Table 1: Number of journals in which each theme appears broken down by gender

Instructor comments related directly to the journals fell into the following categories:

- Recognize the effort turned in based on how time was used.
- Empathize with frustrations.
- Suggest modifications on problem solving process based on how journal indicated tasks were completed (e.g. compiling and testing each class as it is coded, working in stages rather than aiming for A-level immediately).
- Encourage better use of time and/or point out issues of starting late (e.g. suggest taking debugging breaks, note that someone is spending more/less time than most others on the project).
- Reinforce/praise good design and/or debugging habits. (e.g. “Your process seems similar to many other successful programmers and problem solvers.”)

- Encourage student to seek help when a roadblock is encountered.
- Suggest strategies for overcoming roadblocks.
- Reinforce/praise good problem solving strategies (e.g. redesigning a portion of the code that holds bugs you can't find).
- Note progress from struggles indicated by previous journals.
- Suggest strategies for getting started (e.g. writing down all the issues and possible solutions that are occurring to you, then focusing on them one at a time).
- Suggest reference for further textual assistance.

4.2 Discussion

The qualitative analysis indicates that information regularly recorded in PSP documentation is the most common information students record in their journals. Frequently, richer information is also included. Female students are much more likely to mention discussions or assistance and code design than male students.

The combination of the main journal themes and main response themes is indicative of our primary use of the journals as a way to understand how students are going about problem solving related to programming. Students discuss the tasks they complete and the bugs left, including their process to try to solve the bugs. The response to these efforts often suggests further strategies including coming for help. Over time, progress is noted. In reading the journal responses, we also noted that students struggling on the project are more likely to receive detailed feedback relating comments in the journal to specific issues in the code and new strategies.

5 Conclusions and Future Work

The analysis presented here is only a starting point. We do not distinguish between journals of particular projects or between performance quality of the students. It may be that there are distinctions in the type of information based on these categorizations; one would hope to see a more structured and successful approach emerge as students get deeper into the course. This analysis is also incomplete in the sense that we have not looked for conglomerate

information such as points of the project that many people struggled with – whether or not such information can be found is an interesting question. Our data subset consisted of students in the second programming course. We do not know for certain that the same themes emerge in the first programming course. Finally, because our data comes from a single instructor, we do not know if the same sorts of information would be generated when others use this technique.

Nonetheless, this initial study is suggestive for those who are looking for ways to get a deeper sense of what is going through their students heads.

What information do we learn from process journals and how do we use it?

- Information similar to, but less quantified, as PSP. This information is useful and important because we can suggest new strategies for using time and approaching the project and also reinforce positive behavior.
- Information about how the students think about the problem, and specifically where they encounter problems – bugs in thinking and in coding, and how they try to overcome the problems. This information is useful because we gain insight about whether or not the approaches we suggest in class are being used by students, we gain insight into the topics of the course that are difficult, and we gain insight into the particular strengths and weaknesses of individual students. (An example use of this information is the construction of teams in later courses.)
- Information about their emotional response to successes and failures. This information is useful because we can attack one source of retention problems if we can help students develop strategies to deal with frustration. At the same time, we can encourage students who are clearly enjoying the field to explore further.

In some ways one can view the process journal as a less formal, less complete technique that shadows the Personal Software Process. It is a source of fairly rich information with little administration and training cost to the instructor. Its lack of formality encourages acquisition of information that is unlikely to be recorded in a more formal process, but this information is important when evaluating learning rather than training.

We have not introduced PSP methodology to our students at this point, but the analysis suggests that since they are providing similar information, adding this process into their courses could both provide more information and help them be more successful. It would

also be interesting to see if reflective diaries could be used rather than process journals, based around the themes we have pulled out in this study.

The appendix provides four examples of journals and responses. Along with providing a concrete sense of the themes, these examples may suggest further uses for the journal. For example, perhaps a particularly detailed bug analysis could be used as a demonstration of the thought process for other students.

6 Acknowledgments

Students are amazingly honest when asked to keep a journal and we thank them for providing us with insight into their process. We also gratefully acknowledge the support of the National Science Foundation through DUE CCLI A &I Grant 9952548.

References

- [1] Anonymous student process journal, <http://www.cs.xu.edu/~lewadow/journalExample.txt>. Last visited July 16, 2003.
- [2] E. Davis, M. Linn, L. Mann, and M. Clancy. "Mind your Ps and Qs: Using parentheses and quotes in LISP." In C. R. Cook, J. C. Scholtz, and J. C. Spohrer (Eds.), *Empirical Studies of Programmers: Fifth Workshop*, (pp. 62-85). Norwood, NJ: Ablex. 1993.
- [3] A. Fekete, J. Kay, J. Kingston and K. Wimalaratne, "Supporting Reflection in Introductory Computer Science," *Proceedings of the thirty-first SIGCSE technical symposium on computer science education*, pp. 144-148, 2000.
- [4] L. Gugerty, and G. Olson, "Debugging by skilled and novice programmers," CHI'86 Proceedings, pp. 171-174, 1986.
- [5] W. Humphrey *Introduction to the Personal Software Process*. Addison-Wesley, 1997.
- [6] L. Winslow. "Programming Pedagogy – A Psychological Overview," SIGCSE Bulletin, volume 28, No. 3. Sept. 1996, pp. 17-22.

Appendix A: Journal and Response Samples

Below are student journals and instructor responses for four projects over the course of the semester, with some background information as necessary for reading. The examples provide concrete examples of many of the themes seen in journals.

Case 1: Project 0: A simulation with multiple queues. This student starts late and has little success. The response focuses on why spreading the time out will help his process and improve success.

Student Journal

9-6-01

1:00 PM: Startin' now! This is the way I work best. I have about 10 hours till the project is due and I'm in the programmin' groove! Cigarettes and caffeine are the drugs of choice for a day like this! This brings me back to the end of last semester when I was pulling all nighters for <the first programming course>. Right now I'm gonna start by gettin' the input file read, and the loop that'll add processes to the queue, working. I plan on implementing each "process" as a special struct that'll hold the values for the PID, Time arrived, and the Quanta needed. The items pushed onto the stack will be pointers to each of the structs created for each of the processes contained in the data file.

2:25 PM: I have less than 40 lines of code, and a stack of compiler error half that size. I remember why I hate programming now. Oh well. Debugging time.

2:30 PM: So apparently I can't initialize values when I define a stuct. Oops!

3:47 PM: After realizing that all my programming knowledge that I'm almost 4 months removed from wasn't going to just come back to me, I decided to get the old book from <first programming course> from <professor>. This will definately speed up my endeavor. At this point, I have the file reading in, and the total number of processes to be handled calculated. Score one for <me>!

4:10 PM: After conferring with a classmate, I realize that my method would be a real pain in the kiester, and an easier way to do things would be to modify the queue class to handle 3 pieces of data in each queuenode. I really have to learn to start things like this earlier. At this point, I'd be happy with a C if I could simply get this done, working, and finish whatever other work I have for tomorrow. Lesson learned the hard way.

6:58 PM: Back from the cafe, after having taken a break to watch Dragonball, eat food, and clear my head. Now, I really need to get this thang workin' in a manner that will guaruntee me a C.

9:42 PM: How I wished I had started a week ago. Going back to the original Queue of structs idea, with the C Library Queue and not that <expletive> <queue provided to students in first programming course>. I've spent way to much time futzing with that anyway.

11:30 PM: Well, I have the 'D' requirements, as well as a valuable lesson. I will be coming to you for assitance at some point within the next week.

Instructor Response:

Comments:

- 1) Nice log but *ouch*! I started to fear for you as soon as I saw the starting date. I promise you that if you start 5 days ahead of time, minimum, and spend two hours a day (as opposed to the 10 hours in one day) you will do much, much better. The stories about hackers sitting down and working all night are somewhat true, but they largely involve very experienced people doing stuff they know inside and out.
- 2) What comments you have/need are informative (you don't waste anyone's time with comments like "this line does x").
- 3) Your file reading process is ... a bit more complex than

necessary. After you open the file once, you could just read it all in... e.g.

```
queue<process> arrivals; // we store the file data in a queue because
// a) we need some dynamic structure and
// b) we'll be using them in order anyway
while (ProcessList >> proc1.pid >> proc1.arrived >> proc1.quanta)
{
arrivals.push(proc1);
}
```

- 4) Using a struct for the process data is a good idea (I see you debated it a bit in your journal, but it is the best choice so I'm glad you came back to using it).

- 5) *Please* start the next project earlier -- give yourself time to process what needs to be done subconsciously as well as in an adrenaline, caffeine and nicotine rush. To give you some idea: I usually plan meticulously, trying to slow my brain down rather than letting it rush ahead to typing things in -- a very hard thing to resist! Doing it over multiple not-too-long sessions helps find my mistakes. Then I go into the possibly-substance-aided stage of coding. I try to code somewhat as these achievement levels go -- little plateaus of working code rather than the whole thing at once (I learned that the hard way :-)). When I hit bugs, I again back off into shorter sessions -- which sometimes just means I get up and take a walk every half hour.

I'm prattling on like this to you partly because it's good general advice and partly because your code, with multiple file openings and closings where one would have worked, suggests that sometimes your coding brain is racing ahead of your problem-solving brain and you might end up creating more complicated solutions than necessary (which are in turn harder to

debug). Having the luxury of even 12 hours between sessions at the keyboard will do amazing things to fix that imbalance.

Case 2: Project 1: Knight's Tour. A student who's first journal included a suggestion that she start earlier so she could seek help as she hit roadblocks. Because the student has taken the advice to seek help, the instructor comments say less about process (the advice has already been delivered) but praise the improvement and address particular issues that appear in the journal. The process advice given nudges the student towards independence.

Student journal

9-24-01 start time: 10:00 pm end time: 10:45 pm

I wrote the loop that will initialize all the squares to -1. I also read through the assignment again and declared my two dimensional array.

9-29-01 start time: 2:00 pm end time: approximately 5:00 pm

Today, I wrote out the functions that would test to see if a square was empty and legal. (yes those two did take me a while). Then I began to work on finding reachable cells.

9-30-01 start time: 1:00 pm end time: I worked pretty much up until 10:00 pm, however I did have dinner and I also had to take a lifeguarding test which took about 2 hours out of that.

I worked some more on reachable cells, but I could not figure out a way to send the information from that into knightMove. I ended up with a really long and involved function...and then (two hours later) thought of an easier way to do it. Yeah is that not always the case. I also wrote some of main and some of KnightMove, but got stuck when I could not read in the information from reachableCells. So I called it quits for the night.

10-01-01 start time: 2:00 pm end time: 10:00 pm

I went to see <professor> today to see if he could help me to figure out a way to store the cells so that knightmove could use them. Thankfully he had a solution, and after explaining it to me, i then returned to

my cage, i mean dorm room, and continued, now that my road block was cleared. I got pretty much through KnightMove, and began to debug my program. Oy Vey! initialize was fine, but IsLegal and IsEmpty were having quite a few issues. Mainly syntax stuff. I was really dumb with syntax these past couple of days. KnightMove on the other hand--I was getting tons of errors that I had no idea what they meant so I played around with the lines, tried to see if maybe I could fix it. There was one that I could not fix, and guess what it was. Syntax. <professor> caught that one. oops. finished typing main and emailed <professor> what, three times??

10-02-01 start time: 2:30 pm end time: the last possible second
I went to see <professor> again because I had (yes another) syntax error that I couldn't figure out, and of course, he caught it right away. He also suggested a couple of things that would help me avoid errors and that would help to make the program stop. (it's supposed to stop?) Unfortunately, I am getting down to the last possible seconds, and my program doesn't work. It runs, but not correctly. I've talked to a couple of other people and they are having the same issues I am. My program either returns all boards and starting places as having solutions or not having solutions. i figure that i'm moving up in the world however; last semester, my programs did not even compile. So all in all, i'm improving. (can i get credit for that?) So as my microwave chef boyardee and I sit here staring at the computer screen for what feels like hours on end, I don't think I will fix the problem, but i do know (i think) what line it is in. It is in the `if(KnightMove(...)>0) return 1;` because before that, KnightMove does not return anything, so it just skips the if and goes to the else, therefore returning false. I am not sure how to fix that. I don't like to turn in project unfinished, but it looks as if I may have to. My computer screen feels like it is punching me in the eyes. Ok, enough jokes...you can tell i'm getting tired. So, this is project1 signing off.

Instructor response:

1. Good log. It seems to me that your process here is much improved from project 0 which will have positive long-term impact. One more thing to try is to compile as you write so the syntax errors don't stack up quite as high.

2. nice job on function documentation and variable names

3. Oh... a very tiny error in KnightMove:

```
int KnightMove(int BoardSize, int Board[][MAX], int startingRow, int startingColumn,
    move moves[8];
    if (count > BoardSize * BoardSize){
        return 1;
    }
    reachableCells(startingRow, startingColumn, moves);
    for (int i = 0; i < 8; i++){
        if (IsLegal(BoardSize, moves, i) && IsEmpty(Board, moves, i)){
            Board[startingRow][startingColumn] = count;
            count++;
            if (KnightMove(BoardSize, Board, moves[i].row, moves[i].column, count)>0){
return 1;
                }else{
Board[startingRow][startingColumn] = -1;
count--;
return 0;
                }
            }
        }
    }
}
```

That return 0 should be the last line in the function, outside of all the loops. In its current location it returns 0 the very first time a move doesn't work -- preventing it from trying the next move in the list.

other than that, your code looks good. (It's even easy to read.)

4. This feels, to me, like a **vast** improvement over your first project. Keep working, it really is getting better! :-)

Case 3: Project 2: Random Sentence Generator. The student involved had noted late-start issues in earlier journals. This journal is interesting because the student's process is very different from most in that he writes code late in the process and often throws it away. From a performance standpoint, therefore, it looks like he hasn't done any work when he is unsuccessful. The comments praise the think-first strategy as well as his willingness to move on from ideas that are not working.

Student Journal

thursday nov 1 - 11:00 pm = read project description to be able to ask questions on friday

friday nov 2 - 10:30 am = asked a few questions, most importantly found out i was able to do queues of strings

saturday nov 3 - did a good deal of thinking about the project all day

sunday nov 4 - 9:00 pm - 5:00 am = started the actual coding (all of which i rewrote later) got to the i didnt even get to a level level

monday nov 5 - 10:30 am = asked numerous questions in class

11:30 am - 2:15 pm = programed a d level part almost all of which was kept

6:40 pm - 8:20 pm = waited for <professor>, talked to <student> about the project - did some coding did a good amount of coding

8:30 pm - 11:45 pm = used <professor>'s knowledge of c++ to aid in my syntax troubles did a fairly large abmount of coding- threw most out

tuesday nov 6 - 1:00 pm -or so = got more syntax help from <professor>

2:00 pm - 10:15 = worked on roject straight through

got a b version working but my part 1 and 2 both will chop off the first word of the input and i cant figure out what the problem is

11:30 pm = got most of the way - i think to an a grade

but not quite - wont even compile if i leave it as is i had to comment out the lines that call the function, both origanly and recursively,

11:40 pm = will have ran handin on the project

Instructor response:

Comments:

1. Nice journal -- I particularly like that you're not afraid to throw code away. Also it was clear from your questions in class that you gave this project lots of thought.

2. B-level works fine.

3. I don't really follow why the BlankQueue had to be inserted into the hashTable, but otherwise your documentation in that function was fine.

4. I think one of the problems in your outputSentence function is that random() might be very large and you end up dequeuing too much stuff. random() % current.Length() would fix that.

5. The A-part looks pretty good othewise.

6. In general a bit more commenting to guide us through a particular function might be nice -- or names that are a little more clear.

7. Nice effort.

Case 4: Project 3: Room Scheduler. This journal demonstrates great detail of analysis in bugs. The response is typical of cases where the student largely succeeds and no big process issues are noticed.

Student Journal

November 17, 2001

3:00 pm - 4:00 pm

I started this project by writing the Range class. I think there might be problems when I try to output the node of the BST the way I have it written though. The range is supposed to be the key, and the other information is supposed to be stored in data type value. Right now, I have the << of range class outputting "value" also. I'm going to leave it this way until I see if I am for sure wrong the way that I have it.

November 18, 2001

2:30 pm - 4:00 pm

Today I added a class that will store the information about who reserved the room and for what class. I managed to add a function to my main program to add a reservation and then output it correctly.

It took awhile because I had forgotten to close a parenthesis at the end of one of my functions in range.h, but I found the mistake.

November 25, 2001

7:00 pm - 8:00 pm

Tonight I edited my program so that it checks to make sure that the time range requested does not overlap one already reserved. It seems to work. I also added the interface that allows the user to choose which option for the C level. Everything seems to work, although some of the code could probably be cleaned up a bit.

November 26, 2001

9:30 pm - 11:30 pm

Tonight I accomplished a lot on this program. It's to the

B-level!! I cleaned up all of the C-level code, making a lot of it into smaller functions. Then I added the parts to read and output to a file. Everything seems to be working well and the code looks pretty straight-forward and easy to understand. All I still have to work on is deleting a reservation.

11:30 pm - 1:00 am

I was going to quit on this project and work on my calc project, but I decided just to keep going on this because it seemed a lot more interesting. I wrote most of the code for deleting, including the function that determines whether or not the exact node is in the BSTS. The Equal function works, but it doesn't delete the node correctly. I get the "chunk already free" error.

November 27, 2001

10:00 am - 11:15 am

I tried to find my mistake in the Delete function. I started out having running the Equal function and having it return the BST node that matches the desired one by reference. Then I called the normal delete function, having calling search on this BST instead of the root, thinking that it would then find the match on the first try and then be able to go on deleting. Something was wrong with this however, so I totally deleted the line with Search and replaced it with Equal, setting it equal to current, which tells Delete what node to delete. I think there is something wrong with the pointers when I did this, since my Equal function returns a BST, not a pointer to the BST.

November 29, 2001

6:00 pm - 8:00 pm

I gave up trying to straighten out all of my pointers in Delete. I decided to just change my Equal function so it returned a pointer to the BST instead of the actual BST. This straightened everything out very easily. Why didn't I realize that before?

8:30 pm - 10:00 pm

I made my main program shorter by creating a lot of functions, and I was able to eliminate a lot of repeat code by adding parameters to some of the functions so that they would work for both manual and file input/output. Then I added a lot of comments to the program to make what I was doing a lot clearer. It took awhile because when I made the changes to functions, I forgot to pass the BST in by reference, so the changes weren't being recorded. Now I'm done, and I didn't even have to bother you for help like last time!!

Instructor Response:

1. Nice journal. And now I see why it was good that I had Calc before I started taking computer science in college. :-)