Project 7: Processing PDB File Data

1 Overview

For this assignment, you will write a Perl program that can process some of the information contained in the ATOM records in a PDB file whose name is supplied on the command line. Roughly speaking, the program will calculate the frequency of occurrence of various residues and the elements contained in them.

2 Background

The information in this section was provided in Assignment 4. It is repeated in this assignment only for your convenience. A PDB file contains information obtained experimentally, usually by either *X-ray crystallography*, *NMR spectroscopy*, or *cryo-electron microscopy* [1]. (You do not need to know this to do the assignment, but it is important for those who intend to pursue a bioinformatics concentration.) These files completely characterize the molecule, providing, for example,

- the three-dimensional positions of every single atom in the file,
- where the bonds are,
- which amino acids it contains if it is a protein[2] (or nucleotides if DNA or RNA) ,

and much more. The information is not necessarily exact. Associated with some of this information are confidence values that indicate how accurate it is. A PDB file is a plain text file; you can view its contents in any text editor, such as *gedit* or *nedit*, or with commands such as `cat`, `more`, and `less`. Each line in a PDB file begins with a word that characterizes what type of line it is. These individual lines are called *records*. For example, some lines start with the word `REMARK`, which means they are comments about the file itself, or about the experiment through which the data was collected. Some lines start with `SOURCE`, and they have information about the source of the data in the file. Some lines start with words such as `MODEL`, `CONECT`, `ATOM`, and `HETATM`. Each has a different meaning in the file. Take a look at some of the PDB files in the directory `/data/biocs/b/student.accounts/cs132/data/pdb_files` before you read any further, so that you can see what they contain. I suggest picking files that are small, meaning smaller than a megabyte in size.

Proteins are chains of amino acids. Amino acids are organic compounds that carry out many important bodily functions, such as giving cells their structure. They are also instrumental in the transport and the storage of nutrients, and in the functioning of organs, glands, tendons and arteries. Amino acids have names such as *alanine*, *glycine*, *tyrosine*, and *tryptophan*. They are also known more succinctly by unique three-letter codes. The table below lists the twenty standard amino acids with their three-letter codes.

[1] For a summary of what these methods are, see http://www.pdb.org/pdb/static.do?p=education_discussion/Looking-at-Structures/methods.html

[2] Amino acids are the building blocks of proteins, which may contain many thousands of them.

Amino Acid Name	Code
Alanine	Ala
Arginine	Arg
Asparagine	Asn
Aspartate	Asp
Cysteine	Cys
Glutamate	Glu
Glutamine	Gln
Glycine	Gly
Histidine	His
Isoleucine	Ile
Leucine	Leu
Lysine	Lys
Methionine	Met
Phenylalanine	Phe
Proline	Pro
Serine	Ser
Threonine	Thr
Tryptophan	Trp
Tyrosine	Tyr
Valine	Val

Each line that starts with the word `ATOM` represents a unique atom in the protein. So do the lines that start with `HETATM`, but these are atoms in water molecules surrounding the particular protein when it was crystallized, and we want to ignore them for now. Lines that start with `ATOM` contain the three-letter code for the amino acid of which that atom is a part. For example, an atom line for an atom in a phenylalanine molecule looks like this:

```
ATOM 3814 N PHE J 24 -17.763 -7.816 -12.014 1.00 0.00 N
```

The three-letter code is always in uppercase. The exact form of a PDB file is standardized. The standard is revised every few years. The most recent standard that describes the format can be found at:

https://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html

On that page you can scroll down to the `Coordinate Section` and find the link to the `ATOM` record format. There you will see that the line for an atom is defined by the following table:

Remember that Perl stores the words it finds on the command line in the `@ARGV` array and that the first word after the program name is in index 0, not index 1.

- Remember that the `ATOM` records in the PDB file are specified by column positions. You must get the data from the exact set of columns specified above. Therefore, my suggestion is that your program read an entire line and convert it to an array of characters. Then it can use array subscripts to extract the exact subarrays that it needs and convert them back to strings. (Remember that column numbers above start at 1 but arrays start at 0.)
- Remember to follow

COLUMNS	DATA	TYPE	FIELD DEFINITION
1 - 6	Record name	ATOM	
7 - 11	Integer	serial	Atom serial number.
13 - 16	Atom	name	Atom name.
17	Character	altLoc	Alternate location indicator.
18 - 20	Residue name	resName	Residue name.
22	Character	chainID	Chain identifier.
23 - 26	Integer	resSeq	Residue sequence number.
27	AChar	iCode	Code for insertion of residues.
31 - 38	Real(8.3)	x	Orthogonal coordinates for X in Angstroms.
39 - 46	Real(8.3)	y	Orthogonal coordinates for Y in Angstroms.
47 - 54	Real(8.3)	z	Orthogonal coordinates for Z in Angstroms.
55 - 60	Real(6.2)	occupancy	Occupancy.
61 - 66	Real(6.2)	tempFactor	Temperature factor.
77 - 78	LString(2)	element	Element symbol, right-justified.
79 - 80	LString(2)	charge	Charge on the atom.

Notice that the fields are not necessarily separated by white space or by special characters. Instead they are defined by their exact column numbers on the line. For example, the amino acid name is in columns 18 through 20. (The table calls it the ***residue name***.) At this point it would be beneficial to you to make sure that the font you use to view the files is a fixed-width font, so that you can see where the columns are. By default, the terminal in a typical Linux installation uses a *monospace* (fixed-width) font, and so viewing the files using `cat`, `more`, `less`, or any terminal-based editor will be fine.

3 Detailed Requirements

The program should be named `pdbtool.pl`. It must use the file named on the command line as its input file. For example, if the user types

```
pdbtool.pl 1A36.pdb
```

the program must read the data in the file named `1A36.pdb`. If the file argument is missing, or if the file does not exist or cannot be opened for reading, the program should quit with an error message indicating why it quit and providing correct usage. The program should be robust enough that if the file is not a PDB file, it will not crash. This will require no extra effort if the program is designed to look only for the `ATOM` records in the file. The program can assume that all `ATOM` records are in the correct format; it does not have to check that the lines follow that format. The program can assume that the file contains only a single model. If an `ATOM` record contains a non-blank character in column 17, it is an alternate location for an atom already read and should be skipped; the program will use only the first position found.

As the program reads the atom records in the file, it must keep counts of:

- the number of atoms of each element type that it finds,
- the number of atoms of each residue that it finds, and
- the total number of distinct atoms.

When the program has finished reading the file, it should first display, for each distinct element that it found, a line of the form

```
element:   n
```


where `element` is the element name and `n` is the number of atoms of that element. The lines should be sorted alphabetically by element name. Sample output could look like this:

```
C: 3201
N: 918
O: 1101
P: 42
S: 23
...
```

After displaying this information, it should display, for each distinct residue that it found in the file, a line of the form

`residue:` *n*

where `residue` is the three-letter residue name in UPPERCASE and `n` is the number of atoms that were part of the occurrences of that residue in the file, sorted alphabetically. Sample output could like this:

```
ALA: 90
ARG: 44
ASN: 16
ASP: 48
GLN: 45
 ...
```

This would mean that there were 90 atoms in the file occurring in one or more alanine chains, 44 occurring in arginine chains, and so on.

Finally, the program will display a line stating how many atoms were in the file, such as

```
5285 atoms in file 1A36.pdb.
```

4 Programming Considerations

- Remember that Perl stores the words it finds on the command line in the `@ARGV` array and that the first word after the program name is in index 0, not index 1.
- Remember that the `ATOM` records in the PDB file are specified by column positions. You must get the data from the exact set of columns specified above. Therefore, my suggestion is that your program read an entire line and convert it to an array of characters. Then it can use array subscripts to extract the exact subarrays that it needs and convert them back to strings. (Remember that column numbers above start at 1 but arrays start at 0.)
- Remember to follow the Programming Rules document, particularly with respect to proper documentation and proper use of indentation.

5 Testing Your Program

Your program should be thoroughly tested before you submit it. Create some sample small PDB files by editing the ones in the `pdb_files` directory. Manually figure out what the outputs should be or use a spreadsheet to do this if you are good at spreadsheet calculations. Run your program and make sure that your output matches the one you manually computed.

6 Rubric

This homework is graded on a 100 point scale. The program will be graded primarily on its correctness. This means that it does exactly what the assignment states it must do, in detail. Correctness is worth 70% of the grade. Then it is graded on its documentation and design. Comments and appropriate naming of variables are worth 20%; good design another 8%. Naming all files and directories correctly is 2%.

7 Submitting the Program

This assignment is due by the end of the day (i.e. 11:59PM, EST) on Monday, November 19. (I give a grace period of six hours after that, so it is okay to submit it by 6:00 AM of the following day.)

There is a directory in the CSci Department network whose full path name is

`/data/biocs/b/student.accounts/cs132/hwks/hwk7`

You must put it in that directory. To submit your project, you must follow the instructions below exactly! Do not deviate from these instructions.

To be precise:

1. Login using `ssh` to `eniac.cs.hunter.cuny.edu` with your valid username and password, and then `ssh` into any `cslab` host. Do not forget this step. You will not be able to run the `submithwk` command on `eniac`.
2. If you did not do the work on one of the computers in our network, then upload the Perl program file into your home directory. Create a directory named `your-username-hwk7` and put the Perl program into it.
3. Run the command

   ```
   zip -r your-username-hwk7.zip your-username-hwk7
   ```

   This will create the file `your-username-hwk7.zip`. The `zip` command is a special command that compresses the files in the directory and creates a new file that can later be extracted by the `unzip` command. So it will create a "zip file" named `your-username-hwk7.zip` containing your `your-username-hwk7` directory and any files it contains. For example, I would run

   ```
   zip -r sweiss-hwk7.zip sweiss-hwk7
   ```

4. Run the command

   ```
   /data/biocs/b/student.accounts/cs132/bin/submithwk 7  your-username-hwk7.zip
   ```

   Do exactly this. Do not mistype it. The command will create a copy of the file `your-username-hwk6.zip` in the directory

   `/data/biocs/b/student.accounts/cs132/hwks/hwk7`

   It will be named `hwk7_`*`username`*, where *`username`* is your username on the network. You will not be able to read this file, nor will anyone else except for me. If you decide to make any changes and resubmit, just do all the steps again and it will replace the old file with the new one. I will be able to unzip the file, extracting whatever files you created. Do not try to put your file into this directory in any other way - you will be unable to do this.
5. **Do not put anything in this directory other than the file pdbtool.pl. You will lose one point for each file that is in this directory that does not belong there.**

Although these instructions may seem complicated, they simplify the way you submit your work and the way I can retrieve it. If you make mistakes, just start over. If things don't seem to work out, post a question on Piazza with the details included.