CSI 333 – Fall 2011

Programming Assignment III

Administrative Information

- **Team Project**.
- **Deadline:** 11 PM, Friday, Oct. 21, 2011.
  Cutoff: 11 PM, Sunday, Oct. 23, 2011.
- The C source file for the project must be named `p3.c`.
- The source file must be submitted using the `turnin-csi333` command.
- **Each team must make only one submission.**
- README file (on `itsunix.albany.edu`) by 10 PM on Saturday, October 8, 2011.

```
~csi333/public/prog3/prog3.README
```

Administrative Information (continued)

Important Remarks:

- Programs that don't compile or don't generate the executable won't receive any credit.
- Your program must compile and work correctly on `itsunix.albany.edu`.

Lateness Policy:

- No penalty if the program is submitted **by 11 PM on Friday, Oct. 21, 2011**.
- Lateness penalty: 10 points per day.
- Program **won't** be accepted **after 11 PM on Sunday, Oct. 23, 2011**.
- If you submit both a regular version and a late version, only the late version will be graded.

Project Description

Goal: Performing various operations on a linked list.

Weightage: 10% **Total Points:** 100

For students working individually:

Correctness:	85 points
Str. & doc.:	15 points

For students working in a team:

Correctness:	65 points
Str. & doc.:	15 points
Team work:	20 points

Team Projects: Additional Information

- Each team member **must** participate in developing, documenting and testing the program.
- Each team should include additional documentation at the beginning of the source file indicating how the work for the project was divided between the two team members. (Indicate clearly who developed each function and how the testing work was divided between the team members.)
- After the submission deadline, each team must meet with their TA. During the meeting, the TA will ask questions about the team's program and determine the points for team work. (The two team members may receive different scores for team work.)

Information Regarding Linked List

Contents of each node:

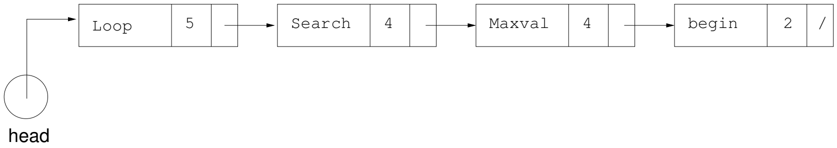
- A string of length at most 10. (This string is called the **symbol** stored in the node.)
- A non-negative integer. (This integer is called the **count** stored in the node.)
- A pointer to the next node of the list.

Properties to be satisfied by the list:

1. The symbols appearing in the list are all distinct; that is, no two nodes have the same symbol.
2. When the list is scanned from left to right, the counts must be in non-increasing order.

Information Regarding Linked List (continued)

Example:



Notes:

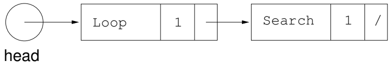
- Initially, the list is empty.
- Your program should process a sequence of commands. The program must continue to accept and execute commands until the user types the `end` command.
- Some commands modify the list while others print information about the list.

List of Commands

- Insert Command: `ins` *str*

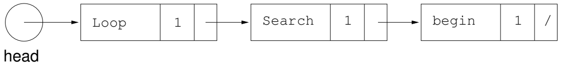
Example 1:

- Current List:



- Command: `ins begin`

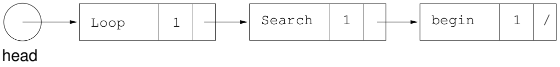
- Resulting List:



Insert Command (continued)

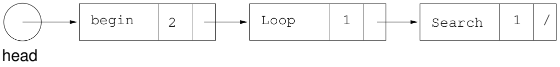
Example 2:

- Current List:



- Command: `ins begin`

- Resulting List:

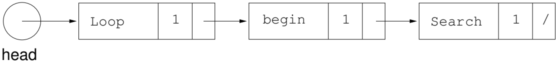


List of Commands (continued)

- Delete Command: `del` *str*

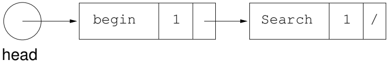
Example 1:

- Current List:



- Command: `del Loop`

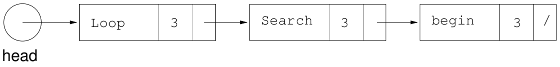
- Resulting List:



Delete Command (continued)

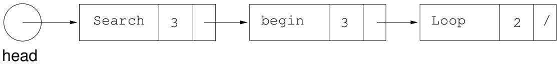
Example 2:

- Current List:



- Command: `del Loop`

- Resulting List:

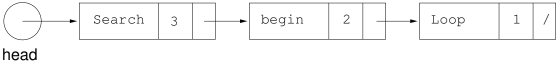


List of Commands (continued)

- Forced Delete Command: `fde` *val*

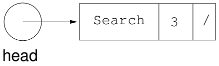
Example:

- Current List:



- Command: `fde 2`

- Resulting List:



List of Commands (continued)

Note: Descriptions of the following commands are given in the handout for this assignment.

- Print Statistics Command: `pst`
- Print List Command: `prl`
- Print using Count Range Command: `pcr` *v1* *v2*

Prefix and Suffix

Prefix: A substring that occurs at the *beginning* of a string. (Each string is a prefix of itself.)

Examples:

- The substring “`val`” is a prefix of the string “`value`”.
- The substring “`Lo`” is a prefix of the string “`Loop`”.

Suffix: A substring that occurs at the *end* of a string. (Each string is a suffix of itself.)

Examples:

- The substring “`ue`” is a suffix of the string “`value`”.
- The substring “`p`” is a suffix of the string “`Loop`”.

List of Commands (continued)

Note: Descriptions of the following commands are given in the handout for this assignment.

- Print Prefix Command: `ppr` *str*
- Print Suffix Command: `psu` *str*
- End Command: `end`

Example of Program Execution

Note: Assume that the executable version of the program is in the file `prog3.out`.

```
unix2> prog3.out

Command?  ins  Loop

Command?  ins  Search

Command?  ins  begin

Command?  prl
   Loop    1
   Search  1
   begin   1

Command?  ins  begin
```

Example of Program Execution (continued)

```
Command?  pcr  2   3
   begin   2

Command?  ins  Loop

Command?  ins  Long

Command?  ppr   Lo
   Loop   2
   Long   1

Command?  ins   Starch

Command?  psu   arch
   Search   1
   Starch   1
```

Example of Program Execution (continued)

```
Command?  pst
   No. of nodes  =  5
   Max. count    =  2
   Min. count    =  1
   Avg. count    =  1.4

Command?  fde  1

Command?  del  begin

Command?  prl
   Loop      2
   begin     1

Command?  end

unix2>
```

Program Outline

```
1. Prompt the user for a command.

2. Read the command.

3. while (command is not "end") {

     (a) Read the value(s) for the command, if any.

     (b) Process the command.

     (c) Prompt the user for the next command.

     (d) Read the next command.

   } /* End of while */
```

Additional Notes

Watch Out!!

- If the `ins`, `del` and `fde` commands (which can modify the list) don't work correctly, the answers for all subsequent commands are likely to be incorrect.
- Test your code thoroughly. If your program crashes during the middle of a sequence of commands, you won't get credit for any of the subsequent commands in that sequence.
- As in the previous programs, use `fflush(stdout)` after each call to `printf`.

Additional Notes (continued)

Remarks:

- Your program must read input from `stdin` and produce all output to `stdout`.
- You may assume that all commands will be error-free; there is no need to deal with erroneous commands.
- If `malloc` returns `NULL`, print an error message and stop.

Structural Requirement: In addition to `main`, you must have a **separate** function to implement each of the commands `ins`, `del`, `fde`, `pst`, `prl`, `pcr`, `ppr` and `psu`.

Suggestions

- Use the `"%s"` format to read the command as a string into a character array of size 4.
- Use the `"%d"` format to read integer values.
- Use the `strcmp` function in the string library (`<string.h>`) to identify which command is specified.
- Use I/O redirection facility of Unix while testing your program.

Program Grading and Other Notes

- Programs will be graded using a script written by the TAs.
- The script will compile your source program, generate the executable version and run the executable on new test data.
- The TAs will grade the version that you submit; once the submission is closed, you won't be allowed to make any changes to your program.
- You must follow the programming and documentation guidelines indicated in "Course Policies".