CS 355 Programming Language Design
Homework 5: Solving 3-SAT problem in C with Abstract Data Structure
Due Monday 04/21/14

1. Introduction

In this homework, you are going to implement a solution for the 3-SAT problem (review homework 2 for the problem detail). The solution will be

- In C programming language,
- Implementing exactly eight steps in homework 2, and
- Implementing a ***list*** Abstract Data Type (ADT) using separate compilation and using the ADT for solving the 3-SAT problem.

The input for your program will be two files; one for the state and one for the clauses.

You will also write up

- The complete Algebraic Specification for the List ADT
- A short reflection on your programming experience with all three programming languages (Scheme, Prolog, and C) in implementing a solution for the 3-SAT problem.

2. List ADT

The list ADT will have the following operations.

- *createList*: a constructor for creating an empty list;
- *isEmpty*: an operation for testing whether or not a list is empty;
- *prepend*: an operation for prepending an entity to a list
- *append*: an operation for appending an entity to a list
- *head*: an operation for determining the first component (or the "head") of a list
- *tail*: an operation for referring to the list consisting of all the components of a list except for its first (this is called the "tail" of the list.)
- *length*: an operation for returning the length of the list
- *insert*: an operation for inserting an element to a specific location in the list
- *remove*: an operation for removing an element at a specific location in the list
- *retrieve*: an operation for getting an element at a specific location in the list

You will need to write the full Algebraic Specification for the list ADT. For implementation, you will need to make it generic. That is, the List ADT does not know what type of its element is and it can be used to make a list of anything. As a result, for some operations that return an element, it also doesn't know what type of the element it is returning is. The implementation should have two separate files: **list.c** and **list.h**.

You will also need to write a program **listaxioms.c** that test all the axioms of the list ADT.

3. 3-SAT in C with List ADT

You will write a program **3sat.c** that implements all 8 steps in homework 2. The input for your program will be either or both of these files: one for the state and one for the clauses. The List ADT must be used to store the state and the clauses in the program memory. The output can be simple printout of the results to the console.

4. What to submit

Please test your code thoroughly before submission. Please create an archive file that contains the following:

- A text file named README that contains the following:
  - Author and contact information (email address);
  - Brief overview of the project including what works and what doesn't work yet.
  - List of instructions necessary for compiling, linking, and executing your program;
  - List of all files in the archive.
  - An Algebraic Specification for the list ADT
  - Your reflection (3+ paragraphs) on your programming experience with all three programming languages. ***In your own opinion***,
    - What are strength and weakness of each language?
    - Which application domain does each language fit the best?
    - Which group of people does each language fit the best?
- All source code and Makefile.
- Any pertinent test files.

This homework will be submitted electronically on Angel. Please double check the files (e.g., download and open them) after you submit. It is due at midnight on the due date.

5. Input Files

5.1. State file (state.txt)

((A #T) (B #F) (C #T) (D #F) (E #T) (F #F) (G #T) (H #T) (I #F) (J #F))

5.2. Clauses file (clauses.txt)

(((NOT I) (NOT H) (NOT F)) ((NOT B) E (NOT G)) (H (NOT C) E)
((NOT G) (NOT E) (NOT I)) ((NOT A) C (NOT J)) ((NOT G) (NOT A) H)
((NOT I) F (NOT H)) (I (NOT A) (NOT E)) ((NOT E) I J) (I (NOT J) (NOT A))
(D I E) ((NOT B) C A) ((NOT G) E I) ((NOT G) (NOT A) F) ((NOT B) A (NOT I))
((NOT G) J E) (D (NOT G) C) (G (NOT B) (NOT C)) ((NOT H) F (NOT J)) (C D F)
(B (NOT G) (NOT E)) (A (NOT D) I) (C A H) (F (NOT B) (NOT A))
((NOT J) (NOT F) (NOT D)) ((NOT A) E (NOT J)) ((NOT I) (NOT G) (NOT C))
(J A C) (H (NOT G) (NOT E)) ((NOT H) (NOT I) (NOT E)) ((NOT H) D (NOT B))
((NOT D) E (NOT I)) (I (NOT G) C) (F (NOT G) (NOT A)) (G I E)
(E (NOT B) (NOT G)) ((NOT C) (NOT I) D) ((NOT G) (NOT B) (NOT H))
(D (NOT I) B) (H I A) ((NOT H) (NOT G) (NOT A)) ((NOT G) (NOT B) F)
(E (NOT G) (NOT A)) (I (NOT H) (NOT A)) ((NOT J) E C) ((NOT B) (NOT A) F)
(G J (NOT A)) ((NOT F) A D) (C D H) (F H E) (E J A) (J A F)
((NOT J) (NOT B) (NOT A)) (E A (NOT J)) ((NOT I) (NOT G) (NOT C))
(I (NOT E) G) (I (NOT A) D) ((NOT F) E (NOT J)) ((NOT H) I D)
(F (NOT B) (NOT H)) (E (NOT H) C) ((NOT F) (NOT D) I) (H (NOT J) (NOT D))
(G (NOT A) (NOT F)) ((NOT E) C J) ((NOT C) E (NOT F)) (D (NOT E) (NOT F))
((NOT H) (NOT A) F) (F E (NOT G)) (D F (NOT J)) ((NOT A) (NOT J) (NOT I))
(J (NOT F) (NOT H)) ((NOT I) (NOT H) J) (C G (NOT E)) (I C F) (I (NOT F) J)
(A (NOT H) (NOT E)) (C (NOT A) (NOT H)) (D (NOT F) (NOT A))
((NOT E) (NOT G) I))