CS3215 (2011): Comments on all the assignments

The project consists of five assignments and prototyping:

- assignment 1: Analysis
- assignment 2: Initial PKB Interface (API) Specifications
- assignments 3-5: three development iterations; in assignment 3 you will also complete PKB Interface Specifications.
- prototyping: done in parallel to assignments 1-2

For each assignment, with exception assignment 5, you will submit a document describing your solution. For assignment 5, you will submit the Final Report. A separate document defines the format for assignments and the Final Report. Please follow these formats. There will be no formal presentation for assignments 1-4. Just bring for the consultation any diagrams, sketches that can help explain your solutions, and to facilitate the discussion and ask questions.

For all the assignments and Final Report, integrate descriptions produced by team members and submit one coherent document per team, organized in the format given in the Required Format for Assignments and Final Project Report.

The Handbook defines the minimum requirements that all teams should meet. Assignments 3-5 show you how to arrive at the final solution incrementally, in three design/iteration steps. You can work at a faster pace if you wish but it is wise to start with the project scope as outlined in assignments and accelerate only once you have got the basics under the control.

Basic expectations and bonus points:

We expect at least implementation of SPA functionality as described in Handbook, reliability (programs must pass our test benchmark), design qualities (flexibility and reusability) and high quality of the project report. Meeting requirements secures B+. To score higher, a team demonstrate some degree of innovation in one or more of the project areas such as SPA features, design solutions, query evaluation strategy, or any other. The project is open-ended. You can extend the scope of the project (and earn extra bonus points) in any area, but first ensure that you have completed the project in the scope described in assignments. Also be sure that extensions you propose are relevant and useful. Plan extensions, if any, for the last iteration and discuss possible extensions with your supervisor.

All team members are expected to equally contribute to the project. We shall conduct peer reviews so that contributions of each team member will be evaluated by other members of his/her team.

Start working on assignments 1-2 and prototyping early. The first four weeks of the course are very intensive. It is most important that all the students in a team are involved in assignments 1-2, and in prototyping: The purpose of assignment 1 is to understand the problem to be solved; the purpose of assignment 2 is to understand the structure of the solution and to set up module interface documentation standards; the purpose of prototyping is to learn C++. Therefore, to effectively contribute to the project, ***all the students in a team must learn what is covered by assignments 1-2 and prototyping.***

CS3215 Assignment 1: Analysis

Due date: Monday, January 24, by 12 noon to your supervisor

Deliverables: Integrate descriptions produced by team members and submit one coherent document per team, organized in the format given in the Required Format for Assignments and Final Project Report.

To get insights into the problem, you will analyze a SIMPLE source program given to you and draw abstract syntax trees (AST), control flow graphs, procedure call table, and other design abstractions. Use your favorite tool to draw diagrams. Then, you will evaluate program queries. During this assignment, you will address questions such as:

- how does an AST look like for a given source program?
- how will I traverse the AST to extract information required for query evaluation?
- you will ask similar questions for CFG and other program design abstractions in the PKB
- which program design abstractions should be stored permanently in the PKB and which should be computed as needed during the query evaluation?
- how do I validate a query? what information do I need to validate a query?
- what does it take to evaluate various types of queries?

At the end of the analysis phase, you should:

- understand the meaning of program design abstractions to be stored in the PKB,
- get an idea of how program design abstractions will be computed,
- get an idea how program design abstractions will be used during query evaluation. For example, what kind of information should be available at AST nodes, how you will traverse the AST or CFG (control flow graph) in order to provide necessary information to answer queries, what it takes to evaluate program queries,
- anticipate technical difficulties and risks,
- collect most of the information needed to complete initial design an SPA architecture (in assignment 2).

Here is the list of activities you are required to do in assignment 1:

1. Study *Part I: Software Requirements* in the Project Handbook.
2. Review the remaining parts of the Project Handbook.
3. Draw an AST for each of the procedures of program Example given below. Indicate Parent and Follows links among the AST nodes.

(continued on the following pages)

```
procedure Example {
1.      x = 1;
2.      y = x;
3.      z = y;
4.      call q1;
5.      i = x;
6.      call q2;
7.      x = z;
8.      while i  {
9.                  x = x - 1;
10.                 if x then {
11.                     x = x+1; }
                    else {
12.                     y = z+x; }
13.                 z = z + x + i;
14.                 call q1;
15.                 i = i - 1; }
16.     call p; }

procedure p {
17.     while i  {
18.                 x = z*3 + 2*y;
19.                 call q2;
20.                 i = i - 1; }
21.     z = z + x + i; }

procedure q1 {
22.     z = x + 1;
23.     x = z + x; }

procedure q2 {
24.     if i then {
25.         z = x + 1; }
        else {
26.         x = z + x; } }
```

Figure 1. Program Example

Evaluate the following queries

Remark: Answer all the queries in the following format:
List the queries and provide query results in the following format:
Select s **such that** Follows*(s, n)
Give answers for n = **1, 4, 9, 15, 19**

n = 1	Ans: none
n = 4	Ans: 5, 6, 7, 8, 16
etc.	

stmt s, s1; assign a; while w; if ifstat; constant c; // declarations of synonyms refer to all the queries listed below

Q 1. **Select** s **such that** Parent (s, s1) **with** s1.stmt#= n
Give answers for n = **3**, **4, 10, 11, 17, 19**

Q 2. **Select** s **such that** Parent (s, n) // what's the difference between query Q1 and Q2?
Give answers for n = **3**, **4, 10, 11, 17, 19**

Q 3. **Select** s **such that** Parent (n, s)
Give answers for n = **3**, **6**, **8**, **10, 19**

Q 4. **Select** w **such that** Parent (w, n)
Give answers for n = **3, 10, 11, 12, 20**

Q 5. **Select** w **such that** Parent* (w, n)
Give answers for n = **3, 10, 11, 12, 20**

Q 6. **Select** s **such that** Follows (s, n)
Give answers for n = **1**, **4, 8, 10, 11, 13, 17, 21**

Q 7. **Select** a **such that** Follows (a, n) // what is the difference between queries Q6 and Q7?
Give answers for n = **1**, **4, 8, 10, 11, 13, 17, 21**

Q 8. **Select** w **such that** Follows (n, w)
Give answers for n = **6, 7**, **16**, **17**.

Q 9. **Select** s **such that** Follows* (s, n)
Give answers for n = **1, 4, 9, 15, 19**

Q 10. **Select** ifstat **such that** Follows* (ifstat, n)
Give answers for n = **4**, **10**, **13**, **15**

Q 11. **Select** a **such that** Follows* (a, n)
Give answers for n = **5, 6, 15, 19, 20, 25**

Q 12. **Select** a **pattern** a ("x", _)

Q 13. **Select** a **pattern** a ("x", _"2*y")

Q 14. **Select** a **pattern** a ("x", _) **such that** Follows (w, a)

Q 15. **Select** s **with** s.stmt# = c.value

Q 16. **Select** BOOLEAN **pattern** ifstat ("i", _, _) **with** c.value = 1

4. Draw a CFG for each procedure in Figure 1.
In your answer, follow example in Project Handbook. Mark CFG nodes with corresponding statement numbers.

Evaluate the following queries

assign a; call c; prog_line n, n1, n2; // declarations of synonyms refer to all the queries listed below

Q 17. **Select** BOOLEAN **such that** Next (n1, n2)
Give answers for the following (n1, n2) values: (**1, 2**), (**6, 8**), (**8, 9**), (**9, 8**), (**10, 12**), (**11, 12**), (**15, 8**), (**16, 17**)

Q 18. **Select** BOOLEAN **such that** Next* (n, n)
Give answers for n = **3**, **8**, **9**, **16**, **17**

Evaluate the following queries

procedure p; variable v; // declarations of synonyms refer to all the queries listed below

Q 19. Select v such that Modifies ("q1", v)

Q 20. Select v such that Modifies (8, v)

Q 21. Select p such that Calls ("Example", p) and Modifies (p, "i") and Uses (p, "y")

Q 22. Select p such that Calls* ("Example", p) and Modifies (p, "z")

5. List variables modified and used in each statement in procedure p and for each procedure in program Example of Figure 1. *Remark*: Follow the definitions of relationships Modifies and Uses given in the handbook. Pay attention to how relationships Modifies and Uses and defined for while and if statements.
6. List variables modified and used in each of the CFG nodes for procedure Example.

For question 5, provide your answer in the following tables:

Statement #	List of modified variables	List of used variables

procedure	List of modified variables	List of used variables
Example		
p		
q		

For question 6, label CFG nodes of procedure Example with unique numbers. Provide your answer in the following table:

CFG node	List of modified variables	List of used variables
n1		
n2		
n3		

Evaluate the following queries

variable v; assign a, a1, a2; while w; if ifstat; // declarations of synonyms refer to all the queries listed below

Q 23. **Select** a **such that** Modifies (a, v)
Give answers for v = **i**, **x**, **y**, **z**

Q 24. **Select** w **such that** Modifies (w, v)
Give answers for v = **i**, **x**, **y**, **z**.

Q 25. **Select** a **such that** Uses (a, v)
Give answers for v = **i**, **x**, **y**, **z**.

Q 26. **Select** v **such that** Uses (n, v)
Give answers for n = **4, 6, 15, 20**

Q 27. **Select** v **such that** Uses (a, v)

Q 28. **Select** a **such that** Modifies (a, v) **and** Uses (a, v)
Give answers for v = **i**, **x**, **y**, **z**.

Q 29. **Select** a **such that** Modifies (a, “x”) **and** Parent (w, a)

Q 30. **Select** a **such that** Parent (w, a) **and** Modifies (a, “x”)

Q 31. **Select** a **such that** Modifies (a, v) **such that** Parent (w, a) **with** v.varName = “x”

Q 32. **Select** BOOLEAN **such that** Affects (a1, a2)
Give answers for the following (a1, a2) values: (**1, 2**), (**1,3**), (**1,4**), (**1,5**), (**1,6**), (**1,7**), (**5, 13**), (**5, 15**), (**7, 9**), (**7, 11**), (**7, 12**), (**7, 13**), (**15, 20**)

Q 33. **Select** a1 **such that** Affects (a1, a2)
Give answers for a2 = **1, 3, 5, 9, 13, 15, 18, 21**

Q 34. **Select** a2 **such that** Affects (a1, a2)
Give answers for a1 = **1, 3, 5, 9, 13**

Q 35. **Select** BOOLEAN **such that** Affects* (a1, a2)
Give answers for the following (a1, a2) values: (**1, 2**), (**1,3**), (**1,4**), (**1,5**), (**1,6**), (**1,7**), (**5, 13**), (**5, 15**), (**7, 9**), (**7, 11**), (**7, 12**), (**7, 13**), (**15, 20**)

Q 36. **Select** a1 **such that** Affects* (a1, a2)
Give answers for a2 = **1, 3, 5, 9, 13, 15, 18, 21**

Q 37. **Select** a2 **such that** Affects* (a1, a2)
Give answers for a1 = **1, 3, 5, 9, 13**

Q 38. **Select** a **such that** Parent (w, a) **and** Modifies (a, “x”)

7. The two queries below yield the same result in SIMPLE. Notice also that this might not be the case in other programming languages. Why?

Select a **pattern** a(“x”, _)
Select a **such that** Modifies (a, “x”)

Assignment 2: Abstract PKB API and SPA Prototype

Due date: Monday, February 7, by 12 noon to your supervisor

Deliverables: Integrate descriptions produced by team members and submit one coherent document per team, organized in the format given in the Required Format for Assignments and Final Project Report.

1. Associations among design abstractions

Describe associations among design abstractions, and explain what they mean and why it is useful to know them.

2. Abstract PKB API

For Assignment 2, you will only document APIs for **VarTable, AST, Modifies** and **Uses**. You will complete abstract PKB API specifications for all the ADTs in the PKB in Assignment 3.

Refer to the examples and recommendations in Handbook Section 9. You can follow the suggested format, or modify some details, if you wish, but be sure that the whole team adopts the same API documentation standard, and that all the ADTs in PKB are documented using the same API documentation standard.

1. Explain naming conventions and other standards adopted in documenting abstract PKB API.
2. Based on the feedback from your supervisor on API specifications submitted in Assignment 2, refine and complete PKB API for all the ADTs described in the Handbook. That is, you should now specify all the ADTs you will be implementing in iterations 1, 2 and 3, not only those you actually implement in iteration 1. Include complete documentation of abstract PKB API in the assignment report.
3. Comment on your abstract PKB API in view of recommendations discussed in Handbook Section 9.

In assignment 3, you will complete PKB API for the remaining ADTs, based on the feedback from your supervisor on your Assignment 2.

Hints:

a) Start by studying **Handbook Section 9** "An architecture of SPA".
b) Be sure that you understand the responsibilities of each SPA component, and PKB.
c) Be sure that you understand the difference between abstract PKB API and concrete PKB API, i.e., class interfaces in C++ program. Consult **Handbook Section 9.4**
d) In **Handbook Sections 9.4-9.6**, you will specific guidelines for discovering and documenting PKB API. Follow these guidelines. Schedule meetings to discuss needs of different SPA components (parser, design extractor, etc.) in respect to various ADTs.
e) Read carefully **Handbook Section 9.7** and be sure you understand guidelines before you start documenting APIs.

3. Parser pseudocode

Write pseudocode for parser of SIMPLE programs, including actions to generate AST during parsing. Study examples provided in Handbook Section "Technical Tips" and extend these examples.

4. Incremental development strategy

Analyze trade-offs in adopting the breath-first vs. depth-first incremental development strategy. Select the strategy and, assuming there will be three iterations in the project, outline in general terms what you would plan to accomplish in each iteration.

Prototyping

Do prototyping together with Assignments 1 and 2.
Due date: Monday, February 7; send zipped code to your supervisor with 1-page instruction how to run it. Demo prototype during consultation.

The idea of the prototype is to let you see all the main SPA components in simplified form. Your will prototype should parse the following subset of *SIMPLE*:
program : procedure+
procedure : 'procedure' proc_name '{' stmtLst '}'
stmtLst : (stmt)+
stmt : assign
assign : var_name '=' var_name | const ';'

The prototype should answer the following queries:
stmt s; stmt# n;

Select s **such that** Follows (n, s)
Select s **such that** Follows* (n, s)
Select s **such that** Follows (s, n)
Select s **such that** Follows* (s, n)
The prototype should allow you to enter a source program, parse it and build an AST, allow you to enter queries and value for statement number n, answer queries, format query result, and display query result. Your parser does not have to deal with error recovery. When your parser encounters the first error, print an error message and stop parsing. Refer to section "Parsing SIMPLE" in "Technical Tips" in Project Handbook.

Your solution should comply to the SPA architecture described in Project Handbook. Even though some of the functionalities are almost trivial, all the functions should be implemented in proper SPA components. Components should communicate through clearly defined interfaces. Adopt the simplest solution for user interface you can think of.

Reminder: Please check on the course web site information regarding libraries that can be used for the project.

--- The End ---