

Department of Computer Science, University of Houston
COSC 6352 - Programming Languages and Paradigms
Fall 2008, Homework – 2, Due Nov. 13 (11.30am) for 1 to 4, Nov. 20 for 5.

1. Explain how classes can be viewed as applications of the three basic principles.
2. Define polymorphism and describe any pragmatic consequences of polymorphic procedure definitions.
3. The cartesian product of two sets A and B is defined as the set $C = \{(a, b) | a \in A, b \in B\}$. In FP write the program “cart2” that takes as input two sets and computes their cartesian product. Also write a Union function that computes the union of two sets.
4. Write rule-based programs for the membership and delete functions for Binary Search Trees. The syntax should be `member(v,T)` and `delete(v,T)` where `v` is a number and `T` is the tree. `member` returns true or false according to whether `v` is in `T` or not. The `delete` function returns `T` if `v` is not in the tree, otherwise it returns the BST obtained by deleting `v` from the tree.
5. Given a rule-based program, `P`, we define a needs graph of the program as follows. The vertices of the needs graph are labeled by the constants and function symbols of `P`. We do not include the built-in operators and constants in constructing this graph. There is a directed edge in the needs graph `**from**` vertex labeled `f` `**to**` vertex labeled `g` provided there is a rule in `P` such that `f` is the top symbol of the left-hand side of this rule and `g` occurs in the right-hand side of this rule. As an example, for the fibonacci rules, there is an edge from vertex labeled `fib` to vertex labeled `if`.

Design and implement in C running under `***Linux***` a program to implement the following task: The input is an equational program in the format attached below. Your program constructs the needs graph of the program and determines the following:

- (a) Whether there is a directed cycle in the needs graph. If yes, the program outputs the labels of the vertices in one cycle.
- (b) Whether there is a directed path containing at least two edges in the needs graph. If yes, the program outputs the labels of the vertices in one such simple path. Simple means that the path does not contain any cycles. The longer the path output the more points.

Points will be given for correctness, documentation and readability, and efficiency (time first, space second). The documentation must include at a minimum a description of all the important data structures and algorithms used. The programs should be resilient against errors, easily modifiable and modular. Extra credit `**may**` be given for innovation and useful extra features.

No partial credit will be given for programs that do not compile on the first attempt. Programs that do not terminate gracefully on any test file for any reason will be graded out of 50 partial credit will not exceed, and in general will be substantially lower than, 50

What to turn in by deadline: Send the files for the programming assignment together in one email: one C source file, one test file you used for your program and the output file for your test file. Send the email with the subject heading `X_Y_2`, where `X` is your last name and `Y` is the last four digits of your ID.

Any paper documents should be securely `**stapled**` together with printed name and social security number, the course number and assignment number on the first page. Deductions

will be made if your documents are unstapled/illegible. Paper documents are due at the ****beginning**** of class. Restrict your documents to no more than 4 standard-size (8.5" X 11") sheets of single-sided single-spaced paper. A printed version of the source code should be attached.

For sample file format check the .m files in the url for lrr given earlier in class.