Problem 1. UML Diagram Interplay (10 pts)

Throughout the semester we have discussed the Unified Modeling Language (UML). UML is known as a *meta-language*. That is, UML may be used to describe itself. As an example, the following UML Class diagram describes the contents of a Statechart diagram.

```
Statechart

State
- name : string(idl)
- entry/action : string(idl)
- do/activity : string(idl)
- event/action : string(idl)
- exit/action : string(idl)

InitState
- name : string(idl) = void
- entry/action : string(idl) = void
- do/activity : string(idl) = void
- event/action : string(idl) = void
- exit/action : string(idl) = void

FinalState
- name : string(idl) = void
- entry/action : string(idl) = void
- do/activity : string(idl) = void
- event/action : string(idl) = void
- exit/action : string(idl) = void

Transition
- name : string(idl) = void
- entry/action : string(idl) = void
- do/activity : string(idl) = void
- event/action : string(idl) = void
- exit/action : string(idl) = void
```

In a previous homework (Hwk5), students were asked to work in groups to determine the interdependence of various UML diagrams. The purpose of this homework problem is to draw a UML Class diagram that describes the entire UML family of diagrams. That is, draw UML as a Class diagram that describes the things, relationships, and dependencies between all UML diagram types.
Problem 2. Activity Diagram (10 pts)

Draw an Activity Diagram for the following C++ operation. The operation fixes a broken invariant in a heap. Specifically, the $k^{th}$ element is heapified in a heap $a[ ]$ of size $N$. Note that it is not necessary to explicitly understand what the code does in order to draw the Activity Diagram. Do not include pre- or post-conditions.

```
Top Down Heapify
void fixDown(Item a[], int k, int N)
  while (2*k <= N)
  {
    int j = 2*k;
    if (j < N && a[j] < a[j+1]) j++;
    if (a[k] >= a[j]) break;
    exch(a[k], a[j]); k = j;
  }
```

- Pass index of array element w/ decreased priority
- Move down the tree
- Exchange the key in the given node with the largest key among the node’s children, moving down to that child, until:
  - we reach bottom of heap
  - there are no children with a larger key
- Unlike root, last node is not known in advance, must pass it.

Problem 3. Dataflow Diagram (10 pts)

Part A) Given the following level 1 Dataflow Diagram, draw the corresponding level 0 Dataflow Diagram. Show all sources and sinks of data. Assume that there is a single process in the level 0 DFD.

Part B) Given the following level 1 Dataflow Diagram, and these additional constraints, draw the level 2 Dataflow Diagram for the SubProc1.2 process. Show all sources and sinks of data.
- SubProc1.2 has 3 ‘sub-subprocs’ named SubProc1.2.1, SubProc1.2.2, and SubProc1.2.3;
- SubProc1.2.1 is the input process, and thus consumes all input to SubProc1.2;
- SubProc1.2.3 is the output process, and thus produces all output from SubProc1.2;
- SubProc1.2.2 is the control process and consumes $df_Z$ and $df_Y$ produced by SubProc1.2.1;
- SubProc1.2.2 also produces $df_X$ that is directly consumed by SubProc1.2.3, and produces $df_W$ that is written to a datastore and eventually consumed by SubProc1.2.1.
Problem 4. Final Exam Study Guide (10 pts)

This problem is submitted individually, separately, and electronically. Formulate a question for the final exam. The question must have two properties: appropriate clarity and appropriate complexity. Clarity means that the description adequately describes the context for the problem. Complexity means that the problem reflects the correct level of difficulty for a senior level, four credit class at the University of Michigan. The set of all problems will be returned ungraded to the students as a study guide for the final exam. Up to three questions from the study guide may be included in the final exam. Submit to eecs486@umich.edu. The question should be in the form: student name, 2 blank lines, question. Please submit as ASCII characters embedded directly into your email submission. Diagrams, if necessary, may be submitted as attachments.