Compilers [Fall 2016]
Test III

NAME: _____

Instructions:

1) This test is 9 pages in length.

2) You have 2 hours to complete and turn in this test.

3) Essays will be graded on how clearly you've communicated the necessary ideas. Write in complete English sentences. Avoid bulleted lists.

4) This test is closed books, notes, papers, friends, neighbors, etc.

5) Write and sign the following: "I pledge my Honor that I have not cheated, and will not cheat, on this test."

_____

_____

Signed: _____

1. [12 points] [Essay]
How does undecidability affect compilers? Describe all the ramifications discussed in class and generalize with Rice's Theorem.

2. [88 points] [Essay] [Your response may continue onto the next 5 pages.]
Let's consider a new language, DJ+, which is DJ with the following four features added:

1. Methods having any natural number of arguments (e.g., "foo()" or "foo(a,b,c)").
2. Arrays. For example, a valid DJ+ program could be

   main { nat[][] a; a=new nat[3][2]; a[1][0]=a[0][1]; }

   DJ+ has exactly 3 expressions for arrays: (1) for allocation, *new T*, where *T* is an array type, (2) for assignment, $e_1[e_2]=e_3$, and (3) for indexing, $e_1[e_2]$ (where the *e*'s are subexpressions) Array sizes must always be positive integers. When allocating memory for an array, the array elements are initialized to their default values.
3. Instanceof expressions, which are used to test dynamically whether an object is an instance of a given class. The format is

   e instanceof C

   where e is a subexpression and C a class name; the whole instanceof expression evaluates to "true" (i.e., a nonzero nat) iff e evaluates to an object whose type is a subtype of C.
4. Switch expressions, as discussed in class. The format is

   switch(e) {n_1 {el_1} n_2 {el_2} … n_i {el_i} default {el_d}}

   where e is an expression, all the n's are natural numbers, all the el's are expression lists, and i is a positive integer. All the natural numbers $n_1..n_i$ must be distinct. Recall that here, control does not "fall through" from one case/branch to another.

How should ***the type-checking and codegen phases*** of dj2dism, implemented as discussed in class, be modified, to make them work for DJ+? Assume the other phases of the dj+2dism compiler are implemented correctly and that DISMs have arbitrarily much code memory available. Use pseudocode but be specific. More efficient implementations, and more efficient target code, will receive more points.

(This page provides extra space for Problem 2.)

(This page provides extra space for Problem 2.)

(This page provides extra space for Problem 2.)

(This page provides extra space for Problem 2.)

(This page provides extra space for Problem 2.)

Undergraduates stop here. The remaining problem is for graduate students.

3. [10 points]
Continuing to think about DJ+ from Problem 2: What's the maximum number of dimensions an allocated array—in a valid DJ+ program P—could have, such that when P is input to dj+2dism, the target code executes on a DISM without errors? Why?