The exam is open book/notes/laptop. We do not guarantee power or Internet access, however.

Answer all 9 questions on the exam paper itself. The total number of points is 180, i.e., one point per minute.

Write your name here: _____________________________________________

I acknowledge and accept the honor code.

(signed) ________________________________________

SCORE TABLE GOES HERE

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
</tr>
</tbody>
</table>
Question 1 (20 pts.)

![Diagram of eight objects and references](image)

The diagram above represents eight objects and the references among them. The following questions involve garbage collection for this network.

a) Any subset of these objects could be in the root set. What is the largest set of objects that could be the root set, and yet there is still at least one object that is garbage?

______________________________

b) Suppose we are using reference counting, and A is the only member of the root set. Suppose also that we delete the references from A to B and from A to D. Which objects are identified as garbage?

______________________________

c) Now suppose we are doing a mark-and-sweep garbage collection with \{A\} as the root set. During the marking phase, we maintain the list of unscanned objects as a queue. When we scan an object with several references, we enqueue those that are in the unreached state in alphabetical order. Give the order in which unscanned objects are dequeued.

______________________________

d) Next, suppose we are using generational garbage collection. None of the eight objects shown are in the root set, and there are no external references to these eight (so they are all garbage). There are two partitions: P0 consists of \{A, B, C, D\}, and P1 consists of \{E, F, G, H\}. Suppose we collect only P0. Which objects are identified as garbage?

______________________________
Question 2 (20 pts.)

Below is a flow graph with one occurrence of an expression $x+y$ and one assignment to $x$. You are going to do the first two steps of the PRE calculation.

(a) Begin with the computation of "Anticipated Expressions." Circle each of the IN or OUT sets that contain $x+y$.

IN(B1) IN(B2) IN(B3) IN(B4) IN(B5)
OUT(B1) OUT(B2) OUT(B3) OUT(B4) OUT(B5)

(b) Now, compute the "Available Expressions" in the sense used for PRE analysis. Circle each of the IN or OUT sets that contain $x+y$.

IN(B1) IN(B2) IN(B3) IN(B4) IN(B5)
OUT(B1) OUT(B2) OUT(B3) OUT(B4) OUT(B5)

_______________________________
**Question 3 (20 pts.)**

Do the following loops have any array data dependences between iterations? If yes, please specify at least one case where the same element of array is accessed across iterations. If no, please explain briefly why not.

(a) 
```java
for (int i = 0 ; i < 100; ++i) {
    A[3*i] = A[3*i + 1];
}
```

**YES**  **NO**

(b) 
```java
for (int i = 0; i <= 1000; ++i) {
    for (int j = 0; j <= 1000; ++j) {
    }
}
```

**YES**  **NO**

(c) 
```java
for (int i = 0; i <= 10; ++i) {
    for (int j = i; j <= 10; ++j) {
    }
}
```

**YES**  **NO**
Question 4 (20 pts.)

Given the following C program,

\[
\begin{align*}
\text{for ( k = 1 ; k <= 100; ++k )} \\
\quad \text{for ( j = 0 ; j <= 150; ++j )} \\
\quad \quad \text{for ( i = j ; i <= 200; ++i )} \\
\quad \quad \quad Z[i][j][k] = i + j + k;
\end{align*}
\]

(a) Please write down its iteration space \(<B, b>\), such that \(B \cdot I + b \geq 0\), where \(0\), or \(0\) with an arrow is a vector of zero's and \(I\) is vector of loop indices. In our code example, \(I\) is \([i, j, k]\) transposed.

\[
\begin{bmatrix}
i \\
j \\
k
\end{bmatrix} + \begin{bmatrix}
i \\
j \\
k
\end{bmatrix} \geq \begin{bmatrix}0\end{bmatrix}
\]

(b) The original program has bad spatial locality because the innermost loop index \(i\) is used as the first array index of \(Z\). Please transform the iteration space (fill in the blanks on the next
NOTE: i, j, and k below are not necessarily the same variables as in the code above.

\[
\begin{align*}
\text{for ( } & k = \_\_\_ \; ; \; k \leq \_\_\_ \; ; \; ++k ) \\
& \text{for ( } j = \_\_\_ \; ; \; j \leq \_\_\_ \; ; \; ++j ) \\
& \text{for ( } i = \_\_\_ \; ; \; i \leq \_\_\_ \; ; \; ++i ) \\
& \quad Z[ \_\_\_ ][ \_\_\_ ][ \_\_\_ ] = i + j + k;
\end{align*}
\]

Question 5 (20 pts.)

Consider the following snippet of Java:

```java
public class Foo {
    public Foo a;
    public Foo b;

    public static void bar(Foo foo1, Foo foo2) {
        if (foo1 == foo2) {
            return;
        }

        foo1.a = foo2;
        baz(foo1);
        baz(foo2);
    }

    public static void baz(Foo foo) {
        foo.b = foo;
    }

    public static void main(String [] args) {
        Foo x = new Foo(); // h1
        Foo y = new Foo(); // h2
        Foo z = new Foo(); // h3

        bar(x, y);
        bar(z, z);
        z.a = x;
    }
}
```
a) What are the \textit{hpts} tuples inferred from this code in a \textbf{context-sensitive} analysis? To get you started, we have given you one of the tuples.

\textit{hpts}(h1, a, h2)

b) What are the \textit{hpts} tuples inferred from this code in a \textbf{context-insensitive} analysis?

c) How many copies of \textit{bar} are made for a context-sensitive analysis? ____________

d) How many copies of \textit{baz} are made for a context-sensitive analysis? ____________

\textbf{Question 6} (20 pts.)

Indicate whether each of the following statements (on the next page) is \textbf{TRUE} or \textbf{FALSE}. Using no more than 3 sentences, explain each answer.
(a) Assuming that no instruction can throw an exception, we are free to move instructions between two control-equivalent blocks of a control-flow graph.

TRUE   FALSE

_______________________________________________________________

_______________________________________________________________

_______________________________________________________________

(b) A single-issue machine will not benefit from techniques like software pipelining or global instruction scheduling because it cannot issue more than one instruction per cycle.

TRUE   FALSE

_______________________________________________________________

_______________________________________________________________

_______________________________________________________________

(c) If we solve a forward dataflow analysis using the maximum fixed-point algorithm, we must initialize all IN[B] to the top element in the lattice.

TRUE   FALSE

_______________________________________________________________

_______________________________________________________________

_______________________________________________________________

(d) For a Boolean function with N inputs, you can always design a truth table such that there are at least $2^{N+1} - 2$ edges in the smallest possible binary decision diagram.

TRUE   FALSE

_______________________________________________________________

_______________________________________________________________

_______________________________________________________________
Question 7 (20 pts.)

In this question, you will write a Datalog analysis that builds on top of the pointer analysis we covered in class. We want to calculate transitive heap connectivity, which is the set of reachable connections between objects in the heap. Specifically, we define a pair of objects $h_1$ and $h_2$ to be in the relation $\text{hpts}^*(h_1,h_2)$ if it is possible, through some number of load instructions on an object represented by $h_1$, to reach an object represented by $h_2$. That is, $\text{hpts}^*$ is the transitive closure of the $\text{hpts}$ relation.

Using the relation $\text{hpts}(h_1,f,h_2)$ that is the output of the pointer analysis, write Datalog rules to calculate the relation $\text{hpts}^*(h_1,h_2)$.

Hint: You will need two rules.

Rule 1: $\text{hpts}^*(h_1,h_2) :- \phantom{\text{hpts}^*(h_1,h_2)}$.

Rule 2: $\text{hpts}^*(h_1,h_2) :- \phantom{\text{hpts}^*(h_1,h_2)}$.

Using the relation $\text{hpts}^*(h_1,h_2)$, we want to find the set of heap objects that can possibly be part of a reference cycle. We define the relation $\text{hpts}^*_{\text{self}}(h)$ to be the set of objects that can possibly be involved in a reference cycle.

Write the Datalog rule to calculate $\text{hpts}^*_{\text{self}}(h)$.

Hint: If you need to, you can use an equality relation “$x = y$” which is true iff $x$ is equal to $y$.

Rule: $\text{hpts}^*_{\text{self}}(h) :- \phantom{\text{hpts}^*_{\text{self}}(h)}$.
If you are using a reference counting garbage collector, objects that are present in the $hpts^*$ relation may leak due to cyclic references.

TRUE   FALSE

If you are using a reference counting garbage collector, objects that are *not* present in $hpts^*_\text{self}$ relation must *not* leak due to cyclic references.

TRUE   FALSE

---

**Question 8** (20 pts.)

From the class, we parallelized a function to sum the $n$ elements of an array $A[i]$. The serial version of the code was

```c
for (int i=0; i<n; i++)
    sum=sum+A[i];
```

Let us assume you have $p$ processors numbered $0...(p-1)$ and a superfast shared memory structure as an array $B[i]$ where $i=0...(p-1)$. Furthermore, assume you write the code once and the same code runs on each processor. However only processor $0$ needs to output the final sum by calling a magic function `return(answer)`. Your code will know which processor it is run on by the magic variable `currentProc` where `currentProc` can be any of $0..(p-1)$. For example, a bad parallel implementation of the code above would be:

```c
for (int i=0; i<n; i++) {
    sum=sum+A[i];
}
B[currentProc]=sum;
if (currentProc==0) return(B[0]);
```
This sums the whole array on each processor, essentially doing $O(n*p)$ steps total. Assuming there is a method `wait()` that blocks until all other threads have finished, a full java implementation of the code snippet above would be:

```java
public int doSum(int[] A, int[] B, currentProc, n, p) {
    int temp=0;
    for (int i=0; i<n; i++) {
        temp=temp+A[i];
    }
    B[currentProc]=temp;
    if (currentProc==0) {
        wait();
        return(B[p-1]);
    } else {return 0};
}
```

The way this normally works is that the main method is run on processor 0 and that method will spawn p Threads, each with a different value of currentProc variable.

**Your question:** fill in the following function to calculate the largest element in the array A (not the sum) with doing $O(n+p)$ steps total. You can assume n is divisible by p:

```java
public int doMax(int[] A, int[] B, currentProc, n, p) {
    int temp = __________;
    for (int i=_______; i<___________); i++ {
        ______________
        ______________
        ______________
    }
    B[_____] = ___________;  
    if (currentProc==0) {
        wait();
        int answer=________;
        for (int i=________; i<__________); i++ {
            ______________
            ______________
            ______________
        }
        return(answer);
    } else {return 0};
}
```
Question 9 (20 pts.)

Suppose $L$ is a semilattice with meet operation $\Lambda$. We can construct a new semilattice $L'$ whose elements are pairs of elements of $L$. We define the meet operation (which we also denote by $\Lambda$) on $L'$ by $[a,b] \Lambda [c,d] = [a \Lambda c, b \Lambda d]$. That is, meet is computed componentwise.

a) Prove that the meet operation on $L'$ is commutative. This proof must be a sequence of equivalent expressions, beginning with $[a,b] \Lambda [c,d]$ and ending with $[c,d] \Lambda [a,b]$. Write this sequence of steps on the lines below, with a justification for each step.

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[a,b] \Lambda [c,d]$</td>
<td>start</td>
</tr>
</tbody>
</table>

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

b) Recall that we say $x \leq y$ if and only if $x \Lambda y = x$. Give an expression for $[a,b] \leq [c,d]$ in the semilattice $L'$ in terms of $\leq$ for the semilattice $L$.

$[a,b] \leq [c,d]$ if and only if __________________________________________________

c) Justify your answer to (b).

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________

____________________________________________________________________________