PL Final

108 points in total

1 a) There are two methods of reclaiming garbage. Which method is used by Perl? How does the method work? (4%)

b) Write a chunk of Perl code to show that a Perl variable may have two values of different types. What is the pro and con of this feature of Perl? (4%)

2 a) Write a chunk of C/C++ code that contains a type error that can't be detected at compile and run time. (4%)

b) Write a chunk of C/C++ code to illustrate that C++ is more strongly typed than C. (4%)

3 a) What are the pros and cons of structure equivalence? (4%)

b) Write a chunk of C/C++ code that uses name equivalence for type checking. Write a chunk of C/C++ code that uses structure equivalence for type checking. (4%)

4 a) In the implementation of static scoping, what are static pointers for? What are dynamic pointers for? (4%)

b) Why are there no static pointers in C/C++ implementations? (4%)

5 a) Consider the following C/C++ function

```
int pow(int a,int n)
{
    if (n==0) return 1;
    else if (n%2==1) return a*pow(a,n-1);
    else return pow(a*a,n/2);     // tail-recursive call
}
int main() { cout << pow(2,3); }
```

Tail-recursively optimize the function **`pow`** by yourself. (A goto version suffices.) (4%)

b) Draw the contents of the C/C++ runtime stack at the point when the recursion reaches its end, i.e. when **`n==0`** becomes true, assuming that the C/C++ compiler does **NOT** perform tail-recursive optimization. (4%)
Be sure to indicate the values of parameters, instruction pointers, and dynamic pointers.

c) Repeat b), but this time assumes that the program is compiled by a C/C++ compiler that does tail-recursive optimization. (4%)

6 Consider the following program written in C++ notation (8%)

```
int k=1,a[2]={1,2};
void p(int x,int y) { x--; y++; cout << a[1]; }
int main()
{
   p(k,a[k]); cout << k << a[0] << a[1];
}
```

What would be the output of this program, if the parameters are passed

a) by reference b) by name c) by value

d) by value-result (where to copy out is determined at calling time)

7 Given the Scheme function

```
(define t
   (lambda (x) (lambda (y) (x (* y y)))))
(define s (lambda (z) (+ z z)))
```

Draw the run-time stack during the evaluation of

```
((t s) (s 2))
```

Highlight the portion of the runtime stack that becomes garbage after the evaluation. (8%)

8 Given the Haskell definitions

```
t x y = x (y*y)
s z = z+z
```

a) Draw the graphs bound to functions **t** and **s**. (4%)

b) Draw the graphs step-by-step during the reduction of (8%)

```
t s (s 2)
```

Hint: There are 8 graphs to draw.

You may ignore the portions of the graphs that become garbage during the reduction.

9 Given the Haskell code

```
ints = enumFrom 1
```

a) Let **facts** = [1,1,2,6,24,…] be an infinite list of factorials, i.e. 0!, 1!, 2!, 3!, 4!, …

Define it in Haskell as a *cyclic* data structure by list comprehension. (4%)

Hint: Use **ints** and **zip**

	facts =	1, **1, 2, 6, 24, 120**, …
×	**ints** =	1, 2, 3, 4, 5, …
		1, 2, 6, 24, 120, …

b) Draw the lazy data structures that represent **`facts`** and **`ints`** after reducing (4%)

```
Hugs> take 4 facts
[1,1,2,6] :: [Integer]
```

Hint: The two lazy data structures are connected. Also, **`facts`** is cyclic, but **`ints`** isn't.

c) (Continuing b)

Suppose we next reduce

1)
```
Hugs> take 3 facts
[1,1,2] :: [Integer]
```

2)
```
Hugs> take 5 facts
[1,1,2,6,24] :: [Integer]
```

How many additions and multiplications are executed in 1)? In 2)? (4%)

10 Given the Prolog program

```
append([],Ys,Ys).
append([X|Xs],Ys,[X|Zs]) :- append(Xs,Ys,Zs).
```

a) Draw the search tree for the goal

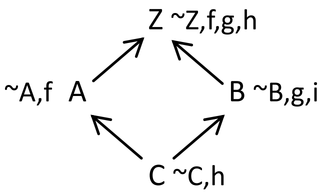
```
?- append(Xs,Ys,[1,2]).
```

Be sure to indicate the solutions found. (8%)

N.B. You may for simplicity rename **`append`** as **`a`**.

b) Use **`append`** to explain why a logic program represents several procedural programs. (4%)

11 Consider the following class lattice in which all functions are virtual



a) Draw a picture showing the structure of an A object. (4%)

b) Draw a picture showing the structure of a C object. (8%)

Note for a) and b) – Wherever it appears, the vtble of a class must have the same layout.