UCLA Computer Science 131 (Fall 2012)
Midterm
100 minutes total, open book, open notes

Name: Preston Chan Student ID: 503943014

1	2	3	4	5	6	7	total
3+3+5	4	16/5	2	2	4	8	52

1. Suppose the following OCaml standard functions were not already predefined. Implement them in Ocaml, or explain why you can't do so. Implement each function without using any other function, except for functions that you define yourself; for example, do not use the builtin function '&' to implement '&&'. Exception: your implementation of '@' can invoke '::'.

1a (3 minutes).
```
val not : bool -> bool  (Boolean negation.)
```

1b (6 minutes).
```
val (&&) : bool -> bool -> bool
  (Conditional 'and'.)
```

1c (6 minutes).
```
val (@) : 'a list -> 'a list -> 'a list
  (List concatenation.)
```

2 (12 minutes). In Homework 2, acceptors are functions that are passed as arguments to matchers. Would it make sense to do it the other way around? That is, to have matchers be functions that are passed as arguments to acceptors? If so, explain how this would work well enough to solve Homework 2; if not, explain why it would not be a good idea. Use code or code snippets to justify your answer.

3a (12 minutes). On the SEASnet GNU/Linux servers, OCaml integer arithmetic uses 63-bit two's complement arithmetic, where integer overflow silently wraps around. For example, at the top level of the interpreter:

```
# 4611686018427387903 + 1;;
- : int = -4611686018427387904
```

Suppose you don't want this behavior. Instead, you want to have an extended integer type with three extra values representing positive infinity, negative infinity, and not-a-number. When the result of a computation falls out of the OCaml int range, you want to substitute an infinite value instead. You want to use not-a-number when the arguments are invalid, just as with IEEE floating point.

Define an OCaml type that will let you represent this extended integer type. Define the operations of addition and subtraction on this type. On overflow, return infinity. When subtracting infinity from infinity, or applying any operation to not-a-number, return not-a-number.

3b (12 minutes). Like (a), but solve the problem for Java. Recall that in Java, Java 'long' values use 64-bit two's complement arithmetic, so that (9223372036854775807L + 1 == -1 - 9223372036854775807L).

3c (12 minutes). Compare the strengths and weaknesses of the two languages in solving the problem as stated.

Preston Chan
503943014

1a. let not = function
| true → false
| false → true

1b. let (&&) a b =
match a with
| false → false
| true → match b with
| false → false
| true → true

8

1c. let rec (@) a b =
match a with
| [] → b
| h::t → [a]::((@) t b)

2. It would not make sense because that would mean you would try to accept the fragment first first, then attempt to match the prefix. This would mean it would go all the way to the end of the grammar before attempting to match any symbol.

4

2. In some sense HW2 already does that, no? It starts by saying "I'm parsing the whole sentence" (i.e., the start symbol).

3a. type ('infinity, 'ninfinity, 'nan, 'regular) integer =
| IN of 'infinity
| NIN of 'ninfinity
| NAN of 'nan
| REG of 'regular

let add a b = match a with
| IN a → IN a
| NAN x → NAN x
| NIN a → NIN a
| REG a → match b with
| INb → INb
| NINb → NINb
| NANb → NANb
| REGb → match a+b with
| result when a<0 && b<0 && result >0 →
NIN result
| result when a>0 && b>0 && result <0 →
IN result
| result → REG result

let sub a b = add a (-b)

3b. public class Extended Integer {
}

```
public class NormalLong implements ExtendedLong {
    private long value;
    public NormalLong(long val){value = val;}
    public long getValue(){
        return value;
    }

    public void setValue(long newVal){
        value = newValue;
    }
}

public class Infinity implements ExtendedLong {}
public class NegativeInfinity implements ExtendedLong {}
public class NotANumber implements ExtendedLong {}

public static ExtendedLong add(ExtendedLong a, ExtendedLong b){
    if (!(a instanceOf NormalLong))
        return a;
    if (!(b instanceOf NormalLong))
        return b;
    NormalLong ca = (NormalLong) a;
    NormalLong cb = (NormalLong) b;
    long potentialResult = ca.getValue() + cb.getValue();
    if ( ca.getValue() > 0 && cb.getValue >0
        && potentialResult < 0)
        return new Infinity();
    } else if (ca.getValue <0 && cb.getValue <0)
            && potential result >0;
            return new NegativeInfinity();
    } else {
        return new NormalLong(potentialValue);
    }
}
```

6 (7 minutes)

4. Consider the following grammar for a simple variant of a subset of C:

(?)

3c. In OCaml, this is much easier because of implicit typing. Java is much harder to use to declare new types as types must be objects ✓ as well.

5

4. Change expr to

expr =
"-" expr

/ "(" expr ")"
/ ALPHA
/ DIGIT

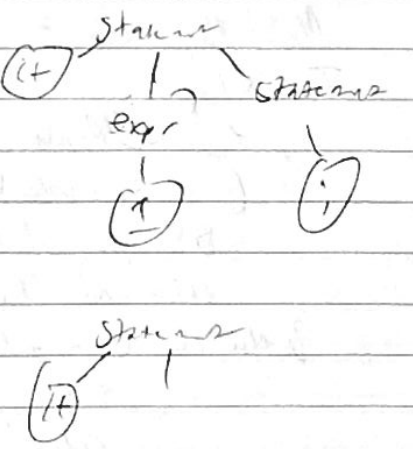
expr1 =
/ expr1 "-" expr

eg. return / -2 - (-3)

2

eg. ide...

could be



5. It isn't a subtype because List<Thread> is not a subtype of List<Object>. If this were a subtype, we would return

2

6. It would make sense if more than one thread is running. An external class could have access to this class and call notify() externally to unblock whatever thread is inside the objects methods. For notify(), another thread could access its mutex, call unlock() and unblock the other thread.

4

7a. $('a \to 'a \to bool) \to ('b \to 'b) \to 'b \to 'a$

2

7b. let rec compute_non_fixed_point eq f x =
if eq (f x) x then compute_non-fixed_point eq f (f x)
else x

6