Instructors: Edmund M. Clarke, Sagar Chaki, Arie Gurfinkel
TAs: Soonho Kong, David Henriques `cmu15414ta@gmail.com`
Due date: 10/05/2011

Assignment 3

Problem 1

Consider the Boolean formula f:

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_4) \wedge (\neg x_3 \vee x_4)$$

Part 1. One possible variable ordering for f is $x_1 < x_2 < x_3 < x_4$. How may possible variable orderings are there for f?

Part 2. Consider two different variable orderings for f, $x_1 < x_2 < x_3 < x_4$ and $x_4 < x_3 < x_2 < x_1$. Draw the BDDs for f for each of the selected variable orderings. For each BDD node v, label its outgoing edge to $low(v)$ with "0" and its outgoing edge to $high(v)$ with "1".

BDD Pseudo-Code Primitives

The following is the pseudo-code for the AND operation on BDDs presented in the class. Recall that we assume a fixed variable ordering that all BDDs must follow.

```
Bdd AND(Bdd f,Bdd g)
{
  if (f == ZERO() || g == ONE())
    return f;
  if (f == ONE() || g == ZERO())
    return g;

  if (VAR(f) == VAR(g))
    return ITE(VAR(f), AND(LOW(f),LOW(g)), AND(HIGH(f),HIGH(g)));

  if (VAR(f) < VAR(g))
    return ITE(VAR(f), AND(LOW(f),g), AND(HIGH(f),g));

  return ITE(VAR(g), AND(LOW(g),f), AND(HIGH(g),f));
}
```

The above pseudo-code introduces the following primitives:

- `==` checks equality between two BDDs.
- `ZERO()` returns the constant "0" BDD. `ONE()` returns the constant "1" BDD.
- `VAR(f)` returns the variable labeling the root of the BDD `f`.

- `LOW(f)` and `HIGH(f)` return the "low" and "high" sub-BDDs of `f`, respectively.
- For two variables `v1` and `v2`, `v1 < v2` iff `v1` appears before `v2` in the variable ordering.
- `ITE(v,f,g)` returns the BDD `h` such that `VAR(h) = v`, `LOW(h) = f`, and `HIGH(h) = g`. In the special case when `f = g`, we have `ITE(v,f,g) = f`.

We will use these primitives in the next problem.

Problem 2

For a BDD f, we write $Formula(f)$ to denote the Boolean formula that f represents. For a Boolean formula Φ and a variable v, and a Boolean value b, we write $\Phi[v = b]$ to mean the Boolean formula obtained by replacing all occurences of v in Φ with b. For example, suppose $\Phi = (\neg x_1 \vee x_2)$. Then $\Phi[x_1 = 1]$ is x_2.

Part 1. Using the primitives introduced earlier, write the pseudo-code for the function `SUB1` that:

1. takes three arguments – a BDD `f`, a variable `v`, and a Boolean value `b`, and
2. returns the BDD `h` such that $Formula(\mathtt{h}) = Formula(\mathtt{f})[\mathtt{v} = \mathtt{b}]$.

In other words, your pseudo-code should look like the following:

```
Bdd SUB1(Bdd f,Var v,Bool b)
{
  ...
}
```

Let Φ be a Boolean formula, and Σ be a conjunction of literals. Then $\Phi \diamond \Sigma$ denotes the formula obtained from Φ as follows:

- for each literal x_i appearing in Σ, replace all occurences of x_i in Φ with "true".
- for each literal $\neg x_i$ appearing in Σ, replace all occurences of x_i in Φ with "false".

Part 2. Using the primitives introduced earlier, write the pseudo-code for the function `SUB2` that:

1. takes two arguments – a BDD `f`, and a BDD `g` such that $Formula(\mathtt{g})$ is a conjunction of literals, and
2. returns the BDD `h` such that $Formula(\mathtt{h}) = Formula(\mathtt{f}) \diamond Formula(\mathtt{g})$.

In other words, your pseudo-code should look like the following:

```
Bdd SUB2(Bdd f,Bdd g)
{
  ...
}
```

Part 3. Using the primitives introduced earlier, write the pseudo-code for the function `IMPL` that:

1. takes two arguments – a BDD `f`, and a variable `v`, and
2. returns "0" if $Formula(\texttt{f}) \Rightarrow \neg\texttt{v}$
3. returns "1" if $Formula(\texttt{f}) \Rightarrow \texttt{v}$
4. returns "-1" otherwise.

In other words, your pseudo-code should look like the following:

```
int IMPL(Bdd f,Var v)
{
  ...
}
```

Assume that `f` is not the "0" BDD. It is OK to add extra helper functions and global variables to your pseudo-code.