CS266 - Formal Specification and Verification
Winter 2009
Homework #2 - Symbolic Execution Homework

Due: Tuesday 20 JAN 09, 11:00am

1. Consider the following program which divides x by y and returns the quotient in quot and the remainder in rem.

```
 1  procedure divide(x,y:integer; var quot,rem:integer);
 2  begin
 3     {: assume ((x>=0) & (y>0)) :}
 4     quot := 0;
 5     rem := x;
 6     {: assert ((x=quot*y + rem) & (rem>=0) & (x=x') & (y=y')) :}
 7     while rem >= y do begin
 8     quot := quot + 1;
 9        rem := rem - y
10     end
11     {: prove ((x'=quot*y' + rem) & (rem>=0) & (rem<y')) :}
12  end;
```

Verify that this program is correct with respect to its input and output assertions as I did for the examples in class. That is, Draw the symbolic execution tree and generate each of the verification conditions (You may prove them if you want.).

2. Choose a program of your liking that contains at least one loop. Your program should be of the same complexity as the factorial example that I presented in class. For this program you are to do the following:

a) Give Entry and Exit assertions that properly describe the function of the program.

b) Give a loop assertion for each loop in the program.

c) Verify that your program is correct with respect to its entry and exit assertions by using the symbolic execution rules that I presented in class. (Draw the symbolic execution tree(s) for the verification of your algorithm.)

If you need any axioms about the integers, such as were needed for the factorial algorithm, you should state the necessary axioms.

3. Give a symbolic execution rule for the Pascal repeat statement.

repeat <statement-list> **until** <Boolean expression>

4. A verification condition is generated for each path in the program that starts at an assertion (i.e., an assume or assert statement) and ends at an assertion (i.e., an assert or prove statement).

Consider the following program which multiplies a by b and returns the product in prod.

```
 1  procedure multiply(a,b:integer; var prod:integer);
 2  var
 3     tema, temb, s: integer;
 4  begin
 5     {: assume (true) :}
 6     tema := a;
 7     prod := 0;
 8     {: assert ((prod=(a-tema)*b) & (a=a') & (b=b')) :}
 9     while tema <> 0 do begin
10        if tema > 0 then
11           s := 1
12        else
13           s := -1;
14        temb := b;
15        {: assert ((prod=(a-tema)*b + s*(b-temb))
                     & (a=a') & (b=b')) :}
16        while temb <> 0 do
17           if temb > 0 then begin
18              prod := prod + s;
19              temb := temb - 1
20           end
21             else begin
22              prod := prod - s;
23              temb := temb + 1
24           end;
25        tema := tema - s
26     end
27     {: prove (prod=a'*b') :}
28  end.
```

For this program I would like you to indicate each of the paths for which a verification condition is generated. The paths should be denoted by a sequence of line numbers (eg., 5,6,7,8).

I would also like you to generate the verification condition for all paths that include line 21.