Kevin James

1.

```
1  method Abs(x : int) returns (y : int)
2    ensures 0 <= y;
3    ensures 0 <= x ==> y == x;
4    ensures x < 0 ==> y == -x;
5  {
6    if (x < 0)c
7     {y := -x;}S1
8    else
9     {y := x;}S2
10 }
```

The post-state, R, is $\{(0 \le x \Rightarrow y == x) \,\&\&\, (x < 0 \Rightarrow y == -x) \,\&\&\, (0 \le y)\}$ (conjunction of ensures clauses)

Using the rule for weakest precondition of an if-else:

$wp(\text{if } c \text{ then } s_1 \text{ else } s_2, R) = [c \wedge wp(s_1, R)] \vee [\neg c \wedge wp(s_2, R)]$

True Branch:

$[c \wedge wp(s_1, R)] = x < 0 \wedge wp(y := -x, 0 \le x \Rightarrow y == x \,\&\&\, x < 0 \Rightarrow y == -x \,\&\&\, 0 \le y)$ $\quad y \leftarrow -x$

$= x < 0 \wedge ((0 \le x \Rightarrow -x == x) \,\&\&\, (x < 0 \Rightarrow -x == -x) \,\&\&\, (0 \le -x))$ by assignment rule

False Branch:

$[\neg c \wedge wp(s_2, R)] = x \ge 0 \wedge wp(y := x, 0 \le x \Rightarrow y == x \,\&\&\, x < 0 \Rightarrow y == -x \,\&\&\, 0 \le y)$ $\quad y \leftarrow x$

$= x \ge 0 \wedge (0 \le x \Rightarrow x == x \,\&\&\, x < 0 \Rightarrow x == -x \,\&\&\, 0 \le x)$ by assignment rule

So now our overall weakest precondition is

$x < 0 \wedge ((0 \le x \overset{x=0}{\Rightarrow} -x == x) \,\&\&\, (x < 0 \Rightarrow -x == -x) \,\&\&\, (0 \le -x)) \vee$

$x \ge 0 \wedge ((0 \le x \Rightarrow x == x) \,\&\&\, (x < 0 \overset{x=0}{\Rightarrow} x == -x) \,\&\&\, (0 \le x))$

which from a quick inspection reduces to True.

Since no requires clause is declared, the pre-state is True. So, we must show that True $\Rightarrow$ True is a tautology. This is so simply by definition of an implication.

2.

```
1 method Q2(x : int, y : int) returns (big : int, small : int)
2   ensures big > small; R
3 {
4   if (x > y) c
5    {big, small := x, y;} s1
6   else
7    {big, small := y, x;} s2
8 }
```

$wp(\text{if } c \text{ then } s_1 \text{ else } s_2, R) = [c \wedge wp(s_1, R)] \vee [\neg c \wedge wp(s_2, R)]$

True Branch:

$[c \wedge wp(s_1, R)] = (x > y) \wedge wp(s_1, big > small)$ ↓ big ← x, small ← y

$= x > y \wedge x > y$ ↓ logical simplification (redundancy)

$= x > y$

False branch

$[\neg c \wedge wp(s_2, R)] = (x \leq y) \wedge wp(s_2, big > small)$ ↓ small ← x, big ← y

$= (x \leq y) \wedge (y > x)$ ↓ logical simplification (both sides are true precisely when $x < y$)

$= x < y$

Putting these together,

$wp(Q2, R) = (x > y) \vee (x < y)$

A. So we need a precondition Q, for which $Q \Rightarrow (x > y) \vee (y > x)$ is a tautology.

In its current form there is no requires clause so the precondition is simply True. This doesn't satisfy the tautology; consider the counterexample $x == y$.

Consider imposing the precondition "requires $x != y$"

B. To prove this is a sufficient precondition, we can show that

$x \neq y \Rightarrow ((x > y) \vee (y > x))$ is a tautology.

When the left side is true, the right side is always true, so this program holds.

3. A.

```
1   method Q3(n0 : int, m0 : int) returns (res : int)
2   ensures res == n0 * m0  Q
3   {
4     var n, m : int;
5     res := 0; S1
6     if (n0 >= 0) C
7       {n,m := n0, m0;} S2
8     else
9       {n,m := -n0, -m0;} S3
10    while (0 < n) E
11    decreases n  D
12    invariant (n*m + res) == n0*m0  I
13    {
14      res := res + m; S4
15      n := n - 1; S5
16    }
17  }
```

Dafny is able to prove this spec

B. I will prove the 5 obligations for total correctness on the following pages.

Proving the 5 obligations for total correctness

1. I holds before the loop. To prove, this we can use wp-calculus on the subset of the method right before the loop, imposing I as a post-condition.

```
30  method subproblem1(n0 : int, m0 : int) returns (res : int)
31  ensures (n*m + res) == n0*m0 Q
32  {
33  |  var n, m : int; Ignore
34  |  res := 0; S1
35  |  if (n0 >= 0)C
36  |  | | |{n,m := n0, m0;}S2
37  |  else
38  |  | | |{n,m := -n0, -m0;}S3
39  }
```

$wp(S, Q) = wp(S_1, wp(\text{if } c \text{ then } S_2 \text{ else } S_3, Q))$

If Else: $= wp(S_1, [c \wedge wp(S_2, Q)] \vee [\neg c \wedge wp(S_3, Q)])$

$[c \wedge wp(S_2, Q)] \vee [\neg c \wedge wp(S_3, Q)] = (n_0 \geq 0 \wedge (n_0 \cdot m_0 + res) == n_0 \cdot m_0) \vee (n_0 < 0 \wedge ((-n_0)(-m_0) + res) == n_0 \cdot m_0)$

Simplify: $= wp(S_1, (n_0 \geq 0 \wedge (n_0 \cdot m_0 + \cancel{res}^{0}) == n_0 \cdot m_0) \vee (n_0 < 0 \wedge ((-n_0)(-m_0) + \cancel{res}^{0}) == n_0 \cdot m_0))$

res := 0: $= (n_0 \geq 0 \wedge (n_0 \cdot m_0) == n_0 \cdot m_0) \vee (n_0 < 0 \wedge ((-n_0)(-m_0)) == n_0 \cdot m_0)$

Simplify: $= (n_0 \geq 0 \wedge \cancel{(n_0 \cdot m_0) == n_0 \cdot m_0}^{\text{True}}) \vee (n_0 < 0 \wedge \cancel{(n_0 \cdot m_0) == n_0 \cdot m_0}^{\text{True}})$

Identity: $= (n_0 \geq 0) \vee (n_0 < 0)$

change representation: $= (n_0 \geq 0) \vee \neg(n_0 \geq 0)$

Law of Excluded Middle: $= \text{True}$

The Requires clause is empty, so R = True, and True ⇒ True is a tautology, so this function is verified, and we prove I holds before the loop. □

2. forall xs, $I \wedge E \Rightarrow wp(S, I)$

```
41  method subproblem2(n0 : int, m0 : int) returns (res : int)
42  requires (n*m + res) == n0*m0 && 0 < n R: I ∧ E
43  ensures (n*m + res) == n0*m0 Q: I
44  {
45      res := res + m; S1
46      n := n - 1; S2
47  }
```

To prove the second obligation, that if the invariant holds before the loop, it will hold after a single iteration of the loop body, we can use wp-calculus on the above subproblem, in which we impose $I \wedge E$ as a precondition, and I as a post-condition. The body of the method is the body of the while loop.

$wp(S, Q) = wp(S_1, wp(S_2, Q))$

$= wp(res := res + m, wp(n := n-1, n \cdot m + res == n_0 \cdot m_0))$ — $n := n-1$

$= wp(res := res + m, (n-1) \cdot m + res == n_0 \cdot m_0)$ — $res := res + m$

$= (n-1) \cdot m + (res + m) == n_0 \cdot m_0$ — simplify

$= m \cdot n - m + res + m == n_0 \cdot m_0$ — simplify

$= m \cdot n + res == n_0 \cdot m_0$

The requires clause is $R = n \cdot m + res == n_0 \cdot m_0 \wedge 0 < n$.

$R \Rightarrow wp(S, Q) =$

$n \cdot m + res == n_0 \cdot m_0 \wedge 0 < n \Rightarrow m \cdot n + res == n_0 \cdot m_0$

The left side of the implication is stronger than the right, so this is a tautology, and so we prove that forall xs, $I \wedge E \Rightarrow wp(S, I)$. □

3. forall xs, $I \wedge \neg E \Rightarrow Q$

So we need to prove

$(n \cdot m + res == n_0 \cdot m_0) \wedge \neg(0 < n) \Rightarrow res == n_0 \cdot m_0$

$= (n \cdot m + res == n_0 \cdot m_0) \wedge (n \leq 0) \Rightarrow res == n_0 \cdot m_0$

Observe in the following 2 proofs, we show that D is lower bounded by 0 during any traversal of the loop body, and we also show that old(D) == D+1 after any single traversal of the while loop. Conjoining these two shows that upon termination of the loop D is lower bounded by -1, so $n > -1$ is a postcondition after the loop. Conjoining this with the antecedent of the implication gives

$(n \cdot m + res == n_0 \cdot m_0) \wedge (n \leq 0) \wedge (n > -1) \Rightarrow res == n_0 \cdot m_0$

$= (n \cdot m + res == n_0 \cdot m_0) \wedge (n == 0) \Rightarrow res == n_0 \cdot m_0$

$= res == n_0 \cdot m_0 \Rightarrow res == n_0 \cdot m_0$

So it is true that forall xs, $I \wedge \neg E \Rightarrow Q$ □

4. forall xs, $I \wedge E \Rightarrow D > 0$

We need to prove that

$(n \cdot m + res == n_0 \cdot m_0) \wedge n > 0 \Rightarrow n > 0$

$= True$

So we prove the decreases expression is bounded below by 0 □

5. forall xs, I ∧ E ⇒ wp(S, old(D)>D)

I will prove the stronger proposition:

forall xs, I ∧ E ⇒ wp(S, old(D)==D+1)

To prove this, we can use wp-calculus to verify the following subproblem, in which the postcondition is old(n)==n+1, the precondition is I ∧ E, and the method body is the loop body.

```
49  method subproblem5(n0 : int, m0 : int) returns (res : int)
50  requires ((n*m + res) == n0*m0) && (n>0)
51  ensures old(n) == n+1
52  {
53    res := res + m;
54    n := n - 1;
55  }
```

wp(S, old(D)==D-1) = wp(S₁, wp(S₂, old(n)==n+1))

n:=n-1 ↓ = wp(S₁, old(n)==(n-1)+1)

res:=res+m ↓ = old(n)==n

now in prestate ↓ = n==n , which is always true.

So, forall xs, I ∧ E ⇒ wp(S, old(D)==D+1)

And this is stronger than our original statement, so we also prove

forall xs, I ∧ E ⇒ wp(S, old(D)>D) □

4. This specification is able to be verified by Dafny.

```
46  method ComputeFact(n : nat) returns (res : nat)
47  requires n > 0;
48  ensures res == fact(n);
49  {
50  |  res := 1;
51  |  var i := 2;
52  |  while (i <= n)
53  |  decreases n - i + 1
54  |  invariant res == fact(i-1) && 1 <= i && i <= n + 1
55  |  {
56  |  |  res := res * i;
57  |  |  i := i + 1;
58  |  }
59  }
60
61  function fact(n : nat): nat
62  requires n > 0;
63  {
64  |  if n == 1 then 1 else n*fact(n-1)
65  }
```

B. Proving the 5 obligations for total correctness

1. I holds before the loop. Proving this is equivalent to verifying the following subproblem

```
67  method subproblem1(n : nat) returns (res : nat)
68  requires n > 0; R
69  ensures res == fact(i-1) && 1 <= i && i <= n + 1; Q
70  {
71  |  res := 1; S1
72  |  var i := 2; S2
73  }
```

$wp(S,Q) = wp(S_1, wp(S_2, Q))$

$= wp(S_1, res == fact(1) \land 1 \leq 2 \land 2 \leq n+1)$ (i:=2, res:=1)

$= 1 == fact(1) \land 1 \leq 2 \land 2 \leq n+1$

$R \Rightarrow wp(S,Q)$

$= n > 0 \Rightarrow 1 == fact(1) \land 1 \leq 2 \land 2 \leq n+1$ (true, true)

$= n > 0 \Rightarrow 2 \leq n+1$ ✓

This is a tautology, so we verify I holds before the loop

2. (forall xs, I ∧ E ⇒ wp(S,I))

```
75   method subproblem2(n : nat) returns (res : nat)
76   requires (res == fact(i-1) && 1 <= i && i <= n + 1) && (i <= n) I ∧ E
77   ensures res == fact(i-1) && 1 <= i && i <= n + 1 I
78   {
79     res := res * i; S1
80     i := i + 1; S2
81   }
```

$wp(S,I) = wp(S_1, wp(S_2, I))$

$i := i+1$ ↓

$= wp(S_1, res == fact(i) \wedge 1 \le i+1 \wedge i+1 \le n+1)$

$res := i*res$ ↓

$= i \cdot res == fact(i) \wedge 1 \le i+1 \wedge i+1 \le n+1)$

$I \wedge E \Rightarrow wp(S,I)$

$= (res == fact(i-1)) \wedge (1 \le i) \wedge (i \le n+1) \wedge (i \le n) \Rightarrow (i \cdot res == fact(i)) \wedge (1 \le i+1) \wedge (i+1 \le n+1)$

So, this tautology holds and we verify $I \wedge E \Rightarrow wp(S,I)$

3. (forall xs, $I \wedge \neg E \Rightarrow Q$)

$(res == fact(i-1) \wedge (1 \le i) \wedge (i \le n+1)) \wedge \neg (i \le n) \Rightarrow res == fact(n)$

$= (res == fact(i-1) \wedge (1 \le i) \wedge (i \le n+1)) \wedge (i > n) \Rightarrow res == fact(n)$

$i == n+1$

$= res == fact(i-1) \wedge 1 \le i \wedge i == n+1 \Rightarrow res == fact(n)$

$= res == fact(n) \wedge 1 \le i \wedge i == n+1 \Rightarrow res == fact(n)$

So, this tautology holds and we verify $I \wedge \neg E \Rightarrow Q$

4. (forall xs, $I \wedge E \Rightarrow D>0$)

$(res == fact(i-1) \wedge (1 \le i) \wedge (i \le n+1)) \wedge (i \le n) \Rightarrow n-i+1 > 0$

So, this tautology holds and we verify $I \wedge E \Rightarrow D>0$

5. (forall xs, $I \wedge E \Rightarrow wp(S, old(D) > D)$)

```
83   method subproblem5(n : nat) returns (res : nat)
84   requires (res == fact(i-1) && 1 <= i && i <= n + 1) && (i <= n) I∧E
85   ensures old(n - i + 1) > n - i + 1  old(D)>D
86   {
87     res := res * i; S1
88     i := i + 1; S2
89   }
```

$wp(S, old(D) > D) = wp(S_1, wp(S_2, old(n-i+1) > (n-i+1)))$ → $n-(i+1)+1 = n-i$

$i := i+1$ ↓ $= wp(S_1, old(n-i+1) > (n-i))$

$res := i*res$ ↓ $= old(n-i+1) > (n-i))$

now in pre-state ↓ $= n-i+1 > n-i$ is always true so the implication is a tautology

So we verify $I \wedge E \Rightarrow wp(S, old(D) > D)$

Thus all 5 obligations are satisfied for total correctness.