Loop Invariants

Selection Sort

The following code implements selection sort on an array A[n]. There are two loops, each of which has a loop invariant.

```
void swap(int&x,int&y)
 {
  int temp = x;
  x = y;
  y = temp;
 }

void sort()
 {
  for(int i = 0; i < n; i++)
   for(int j = i+1; i < n; j++)
    if(A[i] > A[j]) swap(A[i],A[j]);
 }
```

We let Final[i] be the value of A[i] after A is sorted. For ease of understanding, we rewrite the function using while loops.
Outer Loop Invariant: `A[k] = Final[k] for all k < i`
Inner Loop Invariant: `A[ell] >= A[i] for all i < ell < j`

```
void sort()
 {
  int i = 0;
  // Outer Loop Invariant holds
  while (i < n-1)
   {
    // Outer Loop Invariant holds
    int j = i+1;
    // Inner Loop Invariant holds
    while(j < n)
     {
      // Inner Loop Invariant holds
      if{A[j] < A[i])
       swap(A[j],A[i])
      j++
      // Inner Loop Invariant holds
     }
    // Inner Loop Invariant holds
    i++;
    // Outer Loop Invariant holds
   }
  // Outer Loop Invariant holds
 }
```

Doubling

You are working on computer which lacks multipliation and addition. However, it can add or subtract 1 or 2. What does this function do? What is its loop invariant?

```
int double(int n)
 // input condition: n >= 0
 {
  int p = n;
  int q = 0;
  while(p > 0)
   {
    p = p-1;
    q = q+2;
   }
  return q;
 }
```

Real Number to a Non-Negative Integer Power

Recall that $x^0 = 1$ if x is real.[1]
x^a can be computed using recursion.

```
float power(float x, int a)
 // input condition: a >= 0
 {
  if(a == 0) return 1.0;
  else if(a%2) // a is odd
   return x*power(x,a-1);
  else
   return power(x*x,a/2);
 }
```

We prove correctness of this code by strong induction on a.
Proof: By strong induction. Let x be any real number, and let $a \geq 0$ be an integer.
Inductive Hypothesis: `power(x,i)` returns x^i for all $i < a$.
Case 1. $a = 0$. The return value is $1.0 = x^0$.
Case 2: a is odd. By the inductive hypothesis `power(x,a-1)` returns x^{a-1}. Hence `power(x,a)` returns $x \cdot x^{a-1} = x^a$.
Case 3: a is even and greater than zero. `power(x,a)` returns `power(x*x,a/2)` which is equal to $(x^2)^{a/2} = x^a$ by the inductive hypothesis. ∎

We now give a non-recursive version of the function, using a loop invariant to proove correctness. We write `x^a` to mean x^a.

```
1  float power(float x, int a)
2  // input condition: a >= 0
3   {
4    float y = x;
```

[1] What about $x = 0.0^0$? Isn't any power of 0.0 equal to 0.0?

```
 5    int b = a;
 6    float rslt = 1.0;
 7    // Loop invariant: b >= 0 and rslt*y^b == x^a
 8    while(b > 0)
 9     {
10      // Loop invariant: b >= 0 and rslt*y^b == x^a
11      if(b%2) rslt = rslt*y;
12      y = y*y;
13      b = b/2 // truncated integer division
14      // Loop invariant: b >= 0 and rslt*y^b == x^a
15     }
16     // Loop invariant: b >= 0 and rslt*y^b == x^a
17    return rslt;
18   }
```

This code has only one loop. We need to prove that the loop invariant holds before the first iteration of the loop, and that, if it holds at the beginning of an iteration, it holds at the end of that iteration. It follows that the loop invariant holds after the loop exits.

1. LI holds at line 7, because `rslt*y^b == 1.0*x^a`
2. LI holds at line 10 for the first iteration, since it holds at line 7 and none of the variables have been changed.
3. LI holds at line 10 for any other iteration (after the first, that is) since it holds at line 14 of the previous iteration and none of the variables have been changed.
4. Assume LI holds at line 10. We must prove that lI holds at line 14 of the same iteration. This statement is non-trivial, since the values of the variables are changed during the loop.

   To avoid confusion, we use write `rslt, y, b` for the values of those variables at line 10, and `\rslt', y', b'` for the values of the same variables at line 14. Then

   $y' = y^2$

   $b' = \begin{cases} b/2 \text{ if } b \text{ is even} \\ (b-1)/2 \text{ if } b \text{ is odd} \end{cases}$

   $rslt' = \begin{cases} rslt \text{ if } b \text{ is even} \\ rslt \cdot y \text{ if } b \text{ is odd} \end{cases}$

   We know that $b > 0$, since the loop condition must hold. Thus $b' \geq (b-1)/2 \geq 0$.

   If b is even, we have $rslt' \cdot (y')^{b'} = rslt \cdot \left(y^2\right)^{b/2} = rslt \cdot y^b = x^a$ and we are done. If b is odd, we have $rslt' \cdot (y')^{b'} = rslt \cdot y \cdot \left(y^2\right)^{(b-1)/2} = rslt \cdot y \cdot y^{b-1} = rslt \cdot y^b = x^a$ and we are done.
5. Since LI holds at the end of the last iteration, it must hold at line 16 since no values have been changed.

We now prove correctness of the code. Since the loop condition has failed, $b \leq 0$ at line 17. By the loop invariant, $b \geq 0$. Thus, $b = 0$, hence $y^b = 1$. Thus $rslt = rslt \cdot y^b = x^a$ at line 17, so the correct value is returned.
Give a useful loop invariant of each loop. Indicate the places in the code where the invariant holds.

Finding the Minimum of an Array

For this problem, assume that $A[0] \ldots A[n-1]$ is an array of integers, where n is a positive integer.

```
int i = 0;
int j = 0;

while(j < n-1){

  if(A[j] < A[i]) i = j;
  j++;

  }
```

Binary Search

For this problem, assume that $A[0] \ldots A[n-1]$ is a sorted array of integers, where n is a positive integer, and that B is an integer.

```
int lo = 0;
int hi = n;

while(lo < hi){

  int mid = (lo+hi)/2; // truncated division, as in C++
  if(A[mid] < B) lo = mid+1;
  else hi = mid;

  }

if (           ) cout << "Yes" << endl;  // I need to insert a condition here!
else cout << "No" << endl;
```

What do you think the condition of the if statement should be?

Sum of Positive Items

For this problem, assume that $X[0] \ldots X[n-1]$ is an array of float, where n is a positive integer.

```
float sumPositive = 0.0;
int i = 0;
// Loop invariant:

while (i < n){
   // Loop invariant:

   if (X[i] > 0)
    sumPositive += X[i];
   i++;
   // Loop invariant:

  }
assert(i == n);
// Loop invariant:

// We can conclude that sumPositive is the sum of all positive A[i].
cout << sumPositive << endl;
```