Homework 4: 
Loops

Homework due in class on the date indicated above. Late homeworks will be penalized by $5 \cdot 2^{n-1} \%$, where $n$ is the number of days the assignment is late. Please indicate which problems, if any, took extra time.

1. Truth or invariant?

In logic, we know that if $P \rightarrow Q$ ($P$ is stronger than $Q$) and $P$ is true, $Q$ must be true too. Can we say something similar for invariants?

Consider the program

$\text{PROG} \equiv i := 0; \text{do } j < N \rightarrow \text{if } i \leq 0 \rightarrow \text{skip } i > 0 \rightarrow i := i + 1 \text{ fi; } j := j + 1 \text{ od }.$

As usual, our program variables are taken to be integers.

i. Consider the predicate $P \equiv i = 0$. Is it ever made false by $\text{PROG}$? Is it an invariant of the do loop?

ii. Consider the predicate $Q \equiv i \leq 0$. Is it ever made false by $\text{PROG}$? Is it an invariant of the do loop?

iii. Consider the predicate $R \equiv i \geq 1$. Is it ever made false by $\text{PROG}$? Is it an invariant of the do loop?

iv. Comment on the above.

2. A useful function on integers

Consider the function from two integers to one integer $\Gamma = \Gamma(x, y)$ having the following properties:

$\Gamma(x, y) = \Gamma(y, x)$
$\Gamma(x, y) = \Gamma(-x, y)$
$\Gamma(x, y) = \Gamma(x + y, y) = \Gamma(x - y, y)$
$\Gamma(x, y) = |x|$ if $x = y$

i. Define an invariant $P$ and use the given four properties to write a program that manipulates variables $x$ and $y$ such that if it is activated in a state where $x = X \land y = Y \land X > 0 \land Y > 0$, it will terminate with $x = \Gamma(X, Y)$. Generate the correctness proof as you write the program.

ii. The function $t$ we defined in lecture is called a variant function, and it is used to show that a program will terminate in a finite number of steps, or “eventually.” Usually we want to know more than this. If we can introduce an integer function $f$ of the state such that we can show that $f$ is bounded below (e.g., by zero) and that $f$ decreases for each iteration, we can conclude that the program will terminate in at most $f$ iteration steps. (It is clear that any bound function is also a variant function but that only some variant functions are bound functions.) If you didn’t already do so in the previous section, come up with a bound function for how many steps it takes your program to calculate $\Gamma(X, Y)$; the bound function should be given in terms of $X$ and $Y$. Can you say anything about when the performance approaches the bound and when the performance is much better? (The performance is best for $X$ and $Y$ having a very specific relationship.)
iii. Let’s accept as obvious that it follows from $\Gamma(x, y) = \Gamma(x - y, y)$ that $\Gamma(x, y) = \Gamma(x - ny, y)$ for any integer $n$. Use this property to develop a faster implementation of $\Gamma$, using functions $\text{div}$ and $\text{mod}$, if appropriate. As always, state any necessary modifications to the invariant $P$ and show that your program still respects $P$ and that it establishes the necessary postcondition. What is the bound function for the new program?

iv. Let’s assume that the program in the previous part is to be run on a machine that does not actually have built-in hardware to perform $\text{div}$ or $\text{mod}$. Write a second program $\text{DIV}$ that when activated in the state $a = A \land b = B$ terminates with $a = A \div B \land b = A \mod B$. Assume that you do have the operations $+2$ (which is the same as adding a number to itself), $\text{div2}$, $\text{mod2}$ available. As always, your program should be accompanied by a proof of correctness. Also state a bound function.

v. Combine your bound functions to yield an overall count of the number of arithmetic operations it takes to calculate $\Gamma(x, y)$ using an implementation of $\Gamma$ that calls $\text{DIV}$.

vi. Up till now, we have assumed that we can use integers of unlimited range in our programs. This is useful for program development but we know this is not matched by the capabilities of real computers. Assume instead we have to represent our numbers in a base system to base $\beta$, where every variable is limited to the range $[0, \beta - 1]$. As usual a number $x$ will be represented as

$$x = x_0 + x_1\beta + x_2\beta^2 + \cdots = \sum_{i=0}^{N-1} x_i\beta^i.$$ 

To keep things simple, we can assume from now on that $N$ is given as an input. We will represent a number $a$ as an array $a[0], a[1], \ldots, a[N-1]$.

Assume that we are provided functions $+\beta, -\beta$, as well as the relational operators $<\beta, =\beta$, etc. Also assume we have the functions $\text{carry}_\beta(x, y)$ and $\text{borrow}_\beta(x, y)$, which take on the values 0 and 1 only. The functions will operate on the limited-range numbers in $[0, \beta - 1]$ as follows, such that no result is out of $\beta$-range.

$$x + y = x + y + \beta\text{carry}_\beta(x, y)$$
$$x - y = x - y - \beta\text{borrow}_\beta(x, y)$$

Write versions of $+, -, \text{all the relational operators you need}$ for $\text{DIV}$ and your $\Gamma$ program in terms of the operations on limited-range numbers. As always, generate a correctness proof as you program; remember that the correctness proof will have to include a guarantee that all the variables you use stay within the range $[0, \beta - 1]$ at all times—we assume that the $\beta$-operations are equivalent to $\text{abort}$ when attempted on arguments outside that range.

vii. The function $\Gamma$ is of interest in many fields, such as encryption (especially authentication). A busy Web server might have to calculate $\Gamma$ (or a function very much like it) hundreds of times per second. Assume that a particular implementation of 2048-bit encryption runs on a computer of recent manufacture with $\beta = 2^{32}$ and that it takes 1 nanosecond for the computer to perform any of the basic relational or arithmetic operations on $\beta$-range-limited integers. How long does it take this machine, in the worst case, to evaluate $\Gamma(X, Y)$, where $X$ and $Y$ are any integers in the range $[1, 2^{2048} - 1]$ using your first program (using only addition and subtraction)? How long does it take for the program that incorporates $\text{DIV}$?

3. Time management

For 2% assignment credit, keep track of how much time you spend on each problem and turn that in with your other answers.