C241 Homework Assignment 10

1. Performance estimation for recursive programs often involves *recurrence relations* like the one below. Let $a \in \mathbb{N}$. The function $T\colon \mathbb{N} \to \mathbb{N}$ is defined recursively by

$$\begin{aligned} T(0) &= a \\ T(k+1) &= T(k) + k + 1 \end{aligned}$$

We would like to find a *closed form* for T, that is, and an algebraic expression that does not involve recursion

Prove that for all $n \in \mathbb{N}$, $T(n) = a + \dfrac{n^2 + n}{2}$.

2. Using the definition of $\mathcal{V}_3$, check that the expression shown in Example 8.3 evaluates to 17. Then evaluate the following words of L_3:

(a) # $ 3 # 8 9 $ 2 5

(b) # # # $ 3 4 5 6 7

(c) # 3 # 4 # 5 $ 6 7

3. Consider the following language, $L_4 \subseteq V^+$, for $V = \mathbb{N} \cup \{$ `#` , `$` $\}$.

	$L_4 \subseteq V^+$
1.	$\mathbb{N} \subseteq L_4$
2*a*.	$u, v \in L_4 \Rightarrow$ `#` $u\ v \in L_4$
2*b*.	$u, v \in L_4 \Rightarrow u\ v$ `$` $\in L_4$
3.	nothing else

Define an interpretation function for L_4 under which ‘ `#` ’ stands for addition and ‘ `$` ’ for multiplication.

4. Determine whether the following words are in L_4, and if so evaluate them:

(a) # # 2 3 4 $ 4

(b) # 2 3 $ 4 5 $

(c) 2 # 3 # 5 $ 6

(d) 2 # 3 # 5 $ 6

(e) 2 3 # 4 # 5 6 $ 7 $ $

(f) 1 # # 2 3 4 5 6 $ 7 8 $ $ 9 $

5. A language *TERM* of Boolean terms and their meaning $\mathcal{T}$ are defined below. $\mathcal{B} = \langle \{0,1\};\ \wedge, \vee, \neg;\ =;\ 0, 1 \rangle$.

$IVS = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots\}$	$ENV = \{\sigma \mid \sigma \colon IVS \to \{1,0\}\}$
$TERM \subseteq \big(IVS \cup \{\ \mathtt{+}\ ,\ \mathtt{(},\ \mathtt{)}\}\big)^*$	$\mathcal{T} \colon ENV \times TERM \to \{1,0\}$
1*a*. $\mathtt{0} \in TERM$	$\mathcal{T}\sigma\,[\,\mathtt{0}\,] = 0$
1*b*. $\mathtt{1} \in TERM$	$\mathcal{T}\sigma\,[\,\mathtt{1}\,] = 1$
1*c*. $IVS \subseteq TERM$	$\mathcal{T}\sigma\,[\,v\,] = \sigma(v)$ for $v \in IVS$
2*a*. $t \in TERM \Rightarrow \mathtt{-}t \in TERM$	$\mathcal{T}\sigma\,[\,\mathtt{-}t\,] = \neg\mathcal{T}\sigma\,[\,t\,]$
2*b*. $t_1, t_2 \in TERM \Rightarrow \mathtt{(}t_1 \mathtt{+} t_2\mathtt{)} \in TERM$	$\mathcal{T}\sigma\,[\,\mathtt{(}t_1 \mathtt{+} t_2\mathtt{)}\,] = \mathcal{T}\sigma\,[\,t_1\,] \vee \mathcal{T}\sigma\,[\,t_2\,]$
2*b*. $t_1, t_2 \in TERM \Rightarrow \mathtt{(}t_1 \mathtt{*} t_2\mathtt{)} \in TERM$	$\mathcal{T}\sigma\,[\,\mathtt{(}t_1 \mathtt{*} t_2\mathtt{)}\,] = \mathcal{T}\sigma\,[\,t_1\,] \wedge \mathcal{T}\sigma\,[\,t_2\,]$

The *DeMorgan Dual* of $T \in TERM$ is obtained by switching all '0's and '1's; switching all ' + 's and ' * 's; and inserting a ' - ' just before every variable symbol. For instance, the DeMorgan Dual of

```
-( a *  b) + (( c + 0) * ( -b +  a))
```

is

```
-(-a + -b) * ((-c * 1) + (--b * -a))
```

(a) Define a recursive function $\mathcal{D} \colon TERM \to TERM$ that gives the DeMorgan Dual of any $T \in TERM$.

(b) Prove that the DeMorgan Dual of a term is its logical negation:
For all $\sigma \in ENV$ *and* $T \in TERM$, $\mathcal{T}\sigma\,[\,\mathcal{D}[T]\,] = \neg\,\mathcal{T}\sigma\,[\,T\,]$.

6. Chapter 8 briefly outlines of the proof of the Substitution Lemma (Lemma 8.2, p. 150) for the language PROP of propositional formulas (Fig. 8.2, p. 147).

Substution Lemma. Let $\mathcal{S}$ be a substitution and σ an environment. Define a new environment σ' as follows:

$$\sigma'(v) = \mathcal{P}\sigma\left[\mathcal{S}(v)\right] \text{ for } v \in IVS$$

Then for all PROPs P,

$$\mathcal{P}\sigma'\left[P\right] = \mathcal{P}\sigma\left[\mathcal{S}\left[P\right]\right]$$

PROOF. The proof is a straightforward structural induction on PROP. The inductive cases hold because the operations are functions. The interesting base case is the one for a variable $v \in IVS$. In that case, we have

$$\mathcal{P}\sigma'\left[v\right] = \mathcal{P}\sigma\left[\mathcal{S}\left[v\right]\right]$$

which is exactly what we need to make the theorem true. □

Write a detailed outline of the proof, stating all the cases that need to be proved. You do not need to complete the proofs.

7. Let $F \equiv p \wedge (q \vee r)$. Perform the following substitutions

(a) $F \begin{bmatrix} r, q, p \\ p, q, r \end{bmatrix}$

(b) $F \begin{bmatrix} p \vee r \\ p \end{bmatrix}$

(c) $F \begin{bmatrix} p \vee r, & q \Rightarrow r \\ p, & r \end{bmatrix}$

(d) $\left(F \begin{bmatrix} p \vee r \\ p \end{bmatrix}\right) \begin{bmatrix} q \\ p \end{bmatrix}$

(e) $\left(F \begin{bmatrix} q \\ p \end{bmatrix}\right) \begin{bmatrix} p \vee r \\ p \end{bmatrix}$

8. Given a program fragment

$\{\text{PRE}\}$
`while` TEST `do` $\{\text{INV}\}$ BODY
$\{\text{POST}\}$

The *Theorem on Loop Invariants* from Chapter 5 implies that to prove assertion POST holds after the while-loop executes, it suffices to prove three things:

1. INITIALIZATION: PRE $\Rightarrow$ INV.

   Assertion INV must hold when the program first reaches the loop, so whatever condition PRE that does hold must imply POST also holds.

2. INVARIANCE: $\{\text{INV} \wedge \text{TEST}\}$ BODY $\{\text{IMV}\}$

   If assertion INV holds and the loop TEST is true, then after executing the loop BODY, INV will again hold.

3. TERMINATION: INV $\wedge \neg$TEST $\Rightarrow$ INV.

   When (and if) the loop terminates, INV will be true and the loop TEST will have failed. These two conditions should imply that the final assertion, POST, also holds.

In the program to the right, the final assertion states that program variable x holds the greatest common divisor of A and B.

Write down exactly what must be proven to verify this program and do the proofs. If you are not able to finish the proofs, clearly indicate what is left to be proven.

$\{x = A \in \mathbb{N} \wedge y = B \in \mathbb{N}\}$
`while` $x \neq y$ `do`
  $\{\gcd(x, y) = \gcd(A, B)\}$
  `if` $x < y$
    `then` $y :=$ $y - x$
    `else` $x :=$ $x - y$
$\{x = \gcd(A, B)\}$

9. Proposition 8.7 (p. 154) states

$$\{\, P \,\}\ x := t\ \{\, Q \,\} \quad \text{if} \quad P \Rightarrow Q \begin{bmatrix} t \\ x \end{bmatrix}$$

where $Q \begin{bmatrix} t \\ x \end{bmatrix}$ is the formula that results from substituting term t for identifier x in proposition Q.

Use this transformation to reduce

$$\{\, A = qB + r \wedge r < B \,\}\ \texttt{begin}\ q := q + 1\ ;\ r := r - B\ \texttt{end}\ \{\, A = qB + r \,\}$$

to a purely arithmetic (containing no program statements) proposition, as is done in Example 8.8 (p. 156.

SUPPLEMENTAL PROBLEM. (Pill Problem) Once long ago, a man was tried and convicted of a crime, and he was sentenced to death by the Judge. He pleas for mercy. The Judge produces a vial containing twelve pills and a balance scale. He says,

> "Eleven of the pills in this vial are poisonous; you will die within minutes of taking one. One pill is non-poisonous; if you swallow that pill, nothing will happen and you are free to go.
>
> The pills are identical in every other respect, except that the non-poisonous pill has a different weight than any of the poisonous ones.
>
> I will grant you three tries with the balance scale, after which you must take one of the pills.

QUESTION: What are the man's best chances of going free?

ANSWER: If he is smart enough, he will be certain to take the non-poisonous pill and go free."

Explain how.

COMMENT: *To give yourself a real challenge: impose a time limit on solving this problem, and/or don't use pencil and paper (or exhaustive search by computer).*