

## C241 Homework Assignment 6

NOTE: If you cannot finish some of these problems, try at least to “set up” the answer, showing what you would need to do to complete the solution. It is important to demonstrate that you are following a valid approach.

1. Let  $a \in \mathbb{N}$  and define the set  $\{t_0, t_1, \dots, t_k, t_{k+1}, \dots\}$  as follows:

$$\begin{aligned} t_0 &= a \\ &\vdots \\ t_{k+1} &= t_k + k + 1 \end{aligned}$$

- (a) List the first ten values,  $t_0$  through  $t_9$

- (b) Prove by induction on  $n$ : For all  $n \in \mathbb{N}$ ,  $t_n = a + \frac{n^2 + n}{2}$ .

---

SOLUTION
----------

(a)

$$\begin{aligned} t_0 &= a \\ t_1 &= t_0 + 0 + 1 = a + 1 \\ t_2 &= t_1 + 1 + 1 = a + 3 \\ t_3 &= t_2 + 2 + 1 = a + 6 \\ t_4 &= t_3 + 3 + 1 = a + 10 \\ t_5 &= t_4 + 4 + 1 = a + 15 \\ t_6 &= t_5 + 5 + 1 = a + 21 \\ t_7 &= t_6 + 6 + 1 = a + 28 \\ t_8 &= t_7 + 7 + 1 = a + 36 \\ t_9 &= t_8 + 8 + 1 = a + 45 \end{aligned}$$

- (b) The proof is by induction on  $n \in \mathbb{N}$  with hypothesis  $H[k] \equiv t_k = a + \frac{k^2 + k}{2}$ .

BASE CASE.  $t_0 = a = a + \frac{0}{2} = a + \frac{0^2 + 0}{2}$

INDUCTION CASE. Assume  $t_k = a + \frac{k^2 + k}{2}$ . Then

$$t_{k+1} = t_k + k + 1 \quad (\text{definition of } t_*)$$

$$\stackrel{\text{H}}{=} a + \frac{k^2 + k}{2} + k + 1 \quad (\text{I.H.})$$

$$= a + \frac{k^2 + k}{2} + \frac{2(k + 1)}{2} \quad (\text{multiplying } \frac{2}{2} \cdot (k + 1))$$

$$= a + \frac{k^2 + k + 2k + 2}{2} \quad (\text{adding fractions, combining like terms})$$

$$= a + \frac{(k^2 + 2k + 1) + (k + 1)}{2} \quad (\text{rearranging})$$

$$= a + \frac{(k + 1)^2 + (k + 1)}{2} \quad (\text{factoring } k^2 + 2k + 1)$$

Thus  $H[k] \Rightarrow H[k + 1]$ , as needed.

2. Let  $a \in \mathbb{R}$  and define the set  $\{t_0, t_1, t_2, \dots, t_k, t_{k+1}, t_{k+2}, \dots\}$  as follows:

$$\begin{aligned}t_0 &= 0 \\t_1 &= \frac{1}{2} + a \\&\vdots \\t_{k+2} &= 2t_{k+1} - t_k + 1\end{aligned}$$

(a) List the first ten values,  $t_0$  through  $t_9$

(b) Prove: For all  $n \in \mathbb{N}$ ,  $t_n = \frac{n^2}{2} + an$ .

---

SOLUTION
----------

(a)  $t_0 = 0$

$$t_1 = \frac{1}{2} + a$$

$$t_2 = 2t_1 + t_0 + 1 = 2\left(\frac{1}{2} + a\right) + 0 + 1 = 2 + 2a$$

$$t_3 = 2t_2 + t_1 + 1 = \dots = \frac{9}{2} + 3a$$

$$t_4 = 2t_3 + t_2 + 1 = \dots = 8 + 4a$$

$$t_5 = 2t_4 + t_3 + 1 = \dots = \frac{25}{2} + 5a$$

$$t_6 = 2t_5 + t_4 + 1 = \dots = 18 + 6a$$

$$t_7 = 2t_6 + t_5 + 1 = \dots = \frac{49}{2} + 7a$$

$$t_8 = 2t_7 + t_6 + 1 = \dots = 32 + 8a$$

$$t_9 = 2t_8 + t_7 + 1 = \dots = \frac{81}{2} + 9a$$

(b) The proof is by (strong) induction on  $n \in \mathbb{N}$  with hypothesis

$$H[k] \equiv t_k = \frac{k^2}{2} + ak$$

BASE CASE

$$H[0]: t_0 = 0 = \frac{0^2}{2} + a \cdot 0$$

$$H[1]: t_1 = \frac{1}{2} + a = \frac{1^2}{2} + a \cdot 1$$

INDUCTION. Assume that  $t_k = \frac{k^2}{2} + ak$  and  $t_{k+1} = \frac{(k+1)^2}{2} + a(k+1)$ .  
Then

$$\begin{aligned} t_{k+2} &= 2t_{k+1} - t_k + 1 && \text{(by definition)} \\ &\stackrel{\text{IH}}{=} 2 \left[ \frac{(k+1)^2}{2} + a(k+1) \right] - t_k + 1 && H[k+1] \\ &\stackrel{\text{IH}}{=} 2 \left[ \frac{(k+1)^2}{2} + a(k+1) \right] - \left[ \frac{k^2}{2} + ak \right] + 1 && H[k] \\ &= 2 \left[ \frac{(k+1)^2}{2} + \frac{2a(k+1)}{2} \right] - \left[ \frac{k^2}{2} + \frac{2ak}{2} \right] + \frac{2}{2} && \text{(common denom.)} \\ &= \frac{2(k+1)^2 + 4a(k+1) - k^2 - 2ak + 2}{2} && \text{(adding fractions)} \\ &= \frac{(k^2 + 4k + 4) + (2ak + 4a)}{2} && \text{(combining like terms)} \\ &= \frac{(k+2)^2 + 2a(k+2)}{2} && \text{(factoring)} \\ &= \frac{(k+2)^2}{2} + a(k+2) && \text{(simplifying)} \end{aligned}$$

as needed.

3. Use Theorem 4.2 to prove that the program below computes  $A!$ .

```

{x = A}
begin
z := 1;
while x ≠ 1 do {z · x! = A!}
  begin
    z := z * x;
    x := x - 1
  end
end
{z = A!}

```

---

SOLUTION
----------

INITIALIZATION:  $\{x = A\} \ z := 1 \ \{x \cdot x! = A!\}$

$$\begin{aligned}
 z \cdot x! &= z \cdot A! && (\text{precondition, } x = A) \\
 &= 1 \cdot A! && (z := 1) \\
 &= A!
 \end{aligned}$$

INVARIANCE:  $\{(z \cdot x! = A!) \wedge (x \neq 1)\} \ z := z * x \ ; \ x := x - 1 \ \{z \cdot x! = A!\}$   
 Assume that  $z \cdot x! = A!$  and  $x \neq 1$ . Then the loop body assigns  $z' = z \cdot x$  and  $x' = x - 1$ , so

$$\begin{aligned}
 z' \cdot (x')! &= (z \cdot x) \cdot (x - 1)! && (z' = z \cdot x \text{ and } x' = x - 1) \\
 &= z \cdot (x \cdot (x - 1)!) \\
 &= z \cdot x! \\
 &\stackrel{H}{=} A! && (\text{invariance assumption})
 \end{aligned}$$

TERMINATION: By Theorem 4.2 when the program leaves the loop,  $z \cdot x! = A!$  and  $x = 1$ , so

$$A! = z \cdot x! = z \cdot 1! = z$$

Thus, the postcondition is true; the program is correct.

4. **Definition.** The *greatest common divisor* of two whole numbers  $n$  and  $m$ , written  $\gcd(n, m)$ , is the largest number that evenly divides  $n$  and  $m$ .

For example

$$\begin{aligned} \gcd(24, 36) &= 12 \\ \gcd(55, 72) &= 1 \\ \gcd(55, 99) &= 11 \\ \gcd(1, k) &= \gcd(1, k) = 1 \text{ for any } k \\ \gcd(k, \ell) &= \gcd(\ell, k) \text{ for any } k, \ell \in \mathbb{W} \end{aligned}$$

Use Theorem 4.2 to prove that the program below computes  $\gcd(A, B)$ .

```
{A, B ∈ ℤ}
begin
x := A;
y := B;
while (x ≠ y) do {gcd(x, y) = gcd(A, B)}
  begin
  if x < y
    then y := y - x
    else x := x - y
  end
end
{x = gcd(A, B)}
```

More specifically,

- (a) State the *three* things you must prove to verify that the program above computes the greatest common divisor.
- (b) Prove these three things, if you can; or if not, reduce the arguments to *purely mathematical* propositions.

SOLUTION

- (a) 1. INITIALIZATION:

$$\{A, B \in \mathbb{W}\} \quad x := A \ ; \ x := B \quad \{\gcd(x, y) = \gcd(A, B)\}$$

2. INVARIANCE:

$$\begin{aligned} &\{(\gcd(x, y) = \gcd(A, B)) \wedge x \neq y\} \\ &\text{if } x < y \text{ then } y := y - x \text{ else } x := x - y \\ &\{\gcd(x, y) = \gcd(A, B)\} \end{aligned}$$

3. TERMINATION:  $\gcd(x, y) = \gcd(A, B)$  and  $x = y$  imply  $x = \gcd(A, B)$ .

- (b) INITIALIZATION follows immediately (by tautology) from the assignment statements.

TERMINATION follows almost trivially from the fact that  $\text{gcd}(n, n) = n$ .

Proof of INVARIANCE breaks down to two cases, depending on whether  $x < y$ , but the arguments are nearly identical if one observes that  $\text{gcd}(n, m) = \text{gcd}(m, n)$ . That argument reduces to one non-obvious fact:

**Proposition.** If  $x < y$  then  $\text{gcd}(x, y) = \text{gcd}(x - y, y)$ .

COMMENT: Whether or not you can prove this proposition, the most important thing is recognizing that it is the essential insight needed to understand how the program works (and presumably, for writing it in the first place). If you can identify this, you have “solved” the problem.

PROOF: Let  $D = \text{gcd}(x, y)$ . It helps to state exactly what it means to be a “divisor:”

**Definition.** A number  $n$  divides  $m$  iff  $m = p \cdot n$  for some whole number  $p$

According to the definition of  $\text{gcd}$ , there are two things to show:

- (i)  $D$  divides both  $x - y$  and  $y$ .

If  $D$  divides  $x$  then  $x = Dq$  for some  $q$ ; and if  $D$  divides  $y$  then  $y = Dq'$  for some  $q'$ . Thus,

$$x - y = Dq - Dq' = D(q - q')$$

so  $D$  is also a divisor of  $x - y$ .

- (ii) No other common divisor is greater than  $D$ .

Suppose that  $E$  is any divisor of  $x - y$  and  $y$ . That is, for some  $p$  and  $p'$ ,  $x - y = Ep$  and  $y = Ep'$ . Adding these equations together,

$$x - y + y = x = Ep + Ep' = E(p + p').$$

Thus,  $E$  is also a divisor of  $x$ . Since  $D = \text{gcd}(x, y)$  it must be that  $E \leq D$ .

■

5. Prove: If sets  $A$  and  $B$  are countable, then so are  $A \cup B$ ,  $A \times B$  and  $A \setminus B$ .

---

SOLUTION
----------

(a) If  $A$  and  $B$  are countable, then so is  $A \cup B$ .

PROOF: Since  $A$  and  $B$  are countable, there exist injective functions  $C_A: A \rightarrow \mathbb{N}$  and  $C_B: B \rightarrow \mathbb{N}$ . Define  $C_{A \cup B}: (A \cup B) \rightarrow \mathbb{N}$  as follows:

$$C_{A \cup B}(x) = \begin{cases} 2 \cdot C_A(x) & , \text{ if } x \in A \\ 2 \cdot C_B(x) + 1 & , \text{ if } x \in B \setminus A \end{cases}$$

Since it maps elements of  $A$  to even numbers, and elements of  $B$  that are not in  $A$  to odd numbers, and because  $C_A$  and  $C_B$  are injective, there is no possibility that  $C_{A \cup B}$  maps two elements to the same number. In other words,  $C_{A \cup B}$  is an injective function into  $\mathbb{N}$ , which makes  $A \cup B$  countable.

(b) If  $A$  and  $B$  are countable, then so is  $A \times B$ .

PROOF: As above, let  $C_A: A \rightarrow \mathbb{N}$  and  $C_B: B \rightarrow \mathbb{N}$  be injective functions. Let  $C_{\mathbb{Q}}: \mathbb{Q} \rightarrow \mathbb{N}$  be the function defined in Proposition 5.5 and used to show that the Rational Numbers are countable. Define

$$C_{A \times B}(x) = C_{\mathbb{Q}}\left(\frac{C_A(x)}{C_B(x) + 1}\right)$$

(c) If  $A$  and  $B$  are countable, then so is  $A \setminus B$ .

PROOF: Define

$$C_{A \setminus B}(x) = C_A(x)$$

6. Let  $A$  be a finite alphabet. Recall (Defn. 1.9) that  $A^*$  is the set of all words, including the empty word  $\varepsilon$ , over  $A$ .

Is  $A^*$  countable? Explain your reasoning.

---

SOLUTION
----------

$A^*$  is countable.

Let  $|A| = n$ . Imagine listing all the elements of  $A^*$  in the following way:

0. Write down  $\varepsilon$ .

1. Next, list all  $n$  one-letter words in "alphabetical order."

2. Next, list all  $n^2$  two-letter words in "alphabetical order."

⋮

$k$ . Next, list all  $n^k$   $k$ -letter words in "alphabetical order."

and so on forever.

It is evident that listing  $A^*$  in the order described above yields a mapping into  $\mathbb{N}$ , so  $A^*$  is countable by Definition 5.2.

COMMENT. You might want to go further and specify what the index mapping  $C: A^* \rightarrow \mathbb{N}$ . Given an  $k$ -letter word  $w = a_{k-1} \dots a_1 a_0 \in A^*$ ,

(i) The number of shorter words listed prior to  $w$  is  $\sum_{i=0}^{k-1} n^i$ . Note that for  $k=0$ ,  $w = \varepsilon$  and  $C(\varepsilon) = 0$ .

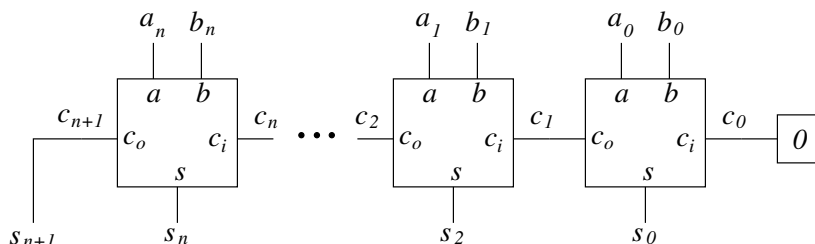
(ii) It remains to determine where in the list of  $k$ -letter words  $w$  is. Since we listed these words in alphabetical order, we can think of  $w$  as a base- $n$  numeral. If  $A = \{x_0, x_1, \dots, x_{n-1}\}$ , interpret letter  $x_i$  as the digit  $i$ . Denote this digit by  $[x]$ . The index associated with  $w = a_{(k-1)} \dots a_0$  is  $\sum_{j=0}^{k-1} [a_k] \cdot n^j$ .

Putting these together,

$$C(a_{(k-1)} \dots a_1 a_0) = \sum_{i=0}^{k-1} n^i + \sum_{j=0}^{k-1} [a_j] \cdot n^j$$

## 7. SUPPLEMENTAL PROBLEM

In Section 2.5, the design is described of a digital circuit for binary addition. An  $n$ -bit adder is an array of  $n$  single-bit full adders



Each full adder, implements the truth table below, which reduces to the system of boolean equations on the right.

	<table border="1" style="border-collapse: collapse; text-align: center;"> <thead> <tr> <th><math>a</math></th> <th><math>b</math></th> <th><math>c_i</math></th> <th style="border-left: 1px solid black;"><math>s</math></th> <th><math>c_o</math></th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td style="border-left: 1px solid black;">0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td style="border-left: 1px solid black;">1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td style="border-left: 1px solid black;">1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td style="border-left: 1px solid black;">0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td style="border-left: 1px solid black;">1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td style="border-left: 1px solid black;">0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td style="border-left: 1px solid black;">0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td style="border-left: 1px solid black;">1</td><td>1</td></tr> </tbody> </table>	$a$	$b$	$c_i$	$s$	$c_o$	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	0	0	1	1	1	1	1	1	$s = a \oplus b \oplus c$ $c = ab + ac + bc$
$a$	$b$	$c_i$	$s$	$c_o$																																											
0	0	0	0	0																																											
0	0	1	1	0																																											
0	1	0	1	0																																											
0	1	1	0	1																																											
1	0	0	1	0																																											
1	0	1	0	1																																											
1	1	0	0	1																																											
1	1	1	1	1																																											

We would like to *prove* that this circuit actually adds. In order to do so, we must first define what *number* an  $n$ -bit binary numeral represents.

**Definition.** A binary numeral  $B = b_n b_{n-1} \cdots b_2 b_1 b_0$  Represents the number

$$\mathcal{N}[B] \stackrel{\text{def}}{=} \sum_{i=0}^n \tilde{b}_i \cdot 2^i \text{ where } \tilde{b} = \begin{cases} \text{the number 0 if } b \text{ is digit 0} \\ \text{the number 1 if } b \text{ is digit 1} \end{cases}$$

Thus, for example,

$$\mathcal{N}[1011] = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 11$$

Using this notation, the property we want to prove is:

**Theorem.** Let  $A \equiv a_{n-1} \cdots a_1 a_0$  and  $B \equiv b_{n-1} \cdots b_1 b_0$  be two  $n$ -bit binary numerals, and let  $S \equiv s_n \cdots s_1 s_0$  be the  $(n+1)$ -bit numeral generated by an  $n$ -bit binary adder with inputs  $A$ ,  $B$ , and  $c_0 = 0$ . Then

$$\mathcal{N}[S] = \mathcal{N}[A] + \mathcal{N}[B]$$

Prove this Theorem.

SOLUTION

SKETCH OF THE PROOF

The proof is by induction on length,  $n \in \mathbb{N}$  of the numerals  $A$  and  $B$ . There is a subtlety, though: we need to prove a slightly more general result that takes the carry bit into account, in order to support the induction:

*Lemma.* Let numerals  $A$ ,  $B$  and  $S$  be as described in the Theorem. Then

$$\mathcal{N}[S] = \mathcal{N}[A] + \mathcal{N}[B] + \tilde{c}_0.$$

(If this lemma holds, then the theorem follows immediately because  $\tilde{0} = 0$ .)

**PROOF OF THE LEMMA.** The proof is by induction on  $n \in \mathbb{N}$ , the length of the numerals  $A$  and  $B$ .

**BASE CASE.** Although it suffices to prove  $H[0]$ , let us also look at  $H[1]$ . According to the truth tables,

$a$	$b$	$c_i$	$c_o$	$s$	$\mathcal{N}[a] + \mathcal{N}[b] + \mathcal{N}[c_i]$	$\stackrel{?}{\cong}$	$\mathcal{N}[c_o s]$
0	0	0	0	0	$0 + 0 + 0$	$\stackrel{\checkmark}{\cong}$	0
0	0	1	0	1	$0 + 0 + 1$	$\stackrel{\checkmark}{\cong}$	1
0	1	0	0	1	$0 + 1 + 0$	$\stackrel{\checkmark}{\cong}$	1
0	1	1	1	0	$0 + 1 + 1$	$\stackrel{\checkmark}{\cong}$	2
1	0	0	0	1	$1 + 0 + 0$	$\stackrel{\checkmark}{\cong}$	1
1	0	1	1	0	$1 + 0 + 1$	$\stackrel{\checkmark}{\cong}$	2
1	1	0	1	0	$1 + 1 + 0$	$\stackrel{\checkmark}{\cong}$	2
1	1	1	1	1	$1 + 1 + 1$	$\stackrel{\checkmark}{\cong}$	3

Thus, the base case holds by inspection of the truth tables.

**INDUCTION.** Assume an  $k$ -bit adder is correct, and consider a  $(k+1)$ -bit adder with inputs  $A = a_k \cdots a_0$ ,  $B = b_k \cdots b_0$ , and  $c_0$ . By the induction hypothesis,

$$\mathcal{N}[s_{k+2} s_k \cdots s_1] = \mathcal{N}[a_{k+1} \cdots a_1] + \mathcal{N}[b_{k+1} \cdots b_1] + \tilde{c}_1$$

Expanding the definition of  $\mathcal{N}[*]$ ,

$$\sum_{i=1}^{k+2} \tilde{s}_i \cdot 2^{i-1} = \sum_{i=1}^{k+1} \tilde{a}_i \cdot 2^{i-1} + \sum_{i=1}^{k+1} \tilde{b}_i \cdot 2^{i-1} + \tilde{c}_1$$

Notice that we are applying the induction hypothesis to the  $k$  higher order bits! To “shift” these, we can multiply both sides of this equation by 2 to get

$$\sum_{i=1}^{k+2} \tilde{s}_i \cdot 2^i = \sum_{i=1}^{k+1} \tilde{a}_i \cdot 2^i + \sum_{i=1}^{k+1} \tilde{b}_i \cdot 2^i + 2 \cdot \tilde{c}_1$$

To complete the induction, one must once again consider all the possible values for  $a_0$ ,  $b_0$  and  $c_0$ , consulting the truth tables to show that in every case,

$$\sum_{i=0}^{k+2} \tilde{s}_i \cdot 2^i = \sum_{i=0}^{k+1} \tilde{a}_i \cdot 2^i + \sum_{i=0}^{k+1} \tilde{b}_i \cdot 2^i + \tilde{c}_0$$

COMMENT. This problem exemplifies a very common pattern of reasoning. A program (or piece of digital hardware) operates on concrete data representing abstract values; and implements some abstract operation. Determining whether the implementation is correct involves the interpretation of the data.

We might have stated the Theorem as: For all  $A, B \in \text{Bit}^n$  and  $c_0 \in \text{Bit}$ ,

$$\mathcal{N}[\text{ADDER}(A, B, 0)] = \mathcal{N}[A] + \mathcal{N}[B]$$

meaning that the interpretation of the ADDER's  $s_i$  output bits equals the arithmetic sum of the interpretations of inputs  $A$  and  $B$ . You will often see theorems of this kind expressed as a "roadmap" diagram like

$$\begin{array}{ccc} \mathbb{N} \times \mathbb{N} & \xrightarrow{+} & \mathbb{N} \\ \mathcal{N} \uparrow & & \uparrow \mathcal{N} \\ \text{Bit}^n \times \text{Bit}^n & \xrightarrow{\text{ADDER}_n} & \text{Bit}^{(n+1)} \end{array}$$