Question 2–1: (Solution, p 5) Complete the following Ada program to read 30 numbers from the user into an array and then echo back to the screen all that exceed 50.

```
with Text_IO; use Text_IO;
procedure Print_Large is
    package Int_IO is new Integer_IO(Integer); use Int_IO;


begin







end Print_Large;
```

Question 2–2: (Solution, p 5) Assume that the Ada program at right compiled successfully.

a. If the compiler assumed pass-by-value parameters, what would this program print?

b. What if it it assumed pass by value-result?

c. What about pass by reference?

```
with Text_IO; use Text_IO;

procedure Test is
    I : Integer;
    J : Integer;

    procedure Y(A : Integer; B : Integer) is
    begin
        A := 5;
        I := 6;
        B := A + I;
    end Y;
begin
    I := 4;
    J := 8;
    Y(I, J);
    Put_Line(Integer'Image(I) & " " & Integer'Image(J));
    Y(I, I); -- Note: I is passed for both parameters!
    Put_Line(Integer'Image(I) & " " & Integer'Image(J));
end Test;
```

Question 2–3: (Solution, p 5) Consider the C code at right.

a. C uses pass-by-value parameters by default. What does this program print?

b. What would it print if it used pass-by-reference parameters by default instead?

c. What about pass-by-value-result parameters?

```
#include <stdio.h>

int p = 5;

void f(int a, int b) {
    a = 2;
    b = 5;
    p = a + p;
}

int main() {
    int a = 3;
    f(a, p);
    printf("%d %d\n", p, a);
    return 0;
}
```

Question 3–1: (Solution, p 5) Use the following grammar in answering the questions below.

$$\mathbf{S} ::= \mathbf{N}\ \texttt{0}\ \mathbf{N}\ \texttt{1}\ \mathbf{N}$$
$$\mathbf{N} ::= \texttt{0}\ \mathbf{N} \mid \texttt{1}\ \mathbf{N} \mid \varepsilon$$

a. Draw a derivation of the sentence “`0 1 0`”.

b. Draw a syntax tree for the sentence “`1 0 1 0`”.

Question 3–2: (Solution, p 6) Consider the following grammar.

$$\mathbf{S} ::= \texttt{a}\ \mathbf{S}\ \texttt{b}\ \mathbf{S} \mid \texttt{x}$$

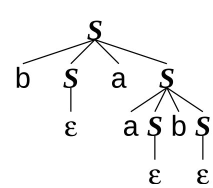
Draw a syntax tree for the sentence

`a a x b x b x`

Question 3–3: (Solution, p 6) Consider the following grammar.

$$\mathbf{S} ::= \varepsilon \mid \texttt{a}\ \mathbf{S}\ \texttt{b}\ \mathbf{S} \mid \texttt{b}\ \mathbf{S}\ \texttt{a}\ \mathbf{S}$$

a. Give a derivation for the sentence `b a a b`. The following tree proves that it is possible.



b. Give a derivation for the sentence `a b a a b b`.

c. Draw a parse tree for the sentence `a b a b`. (See part (a) for an example of how to indicate a derivation to the empty sequence.)

d. Give a succinct English description of the language that this grammar describes. This description should *not* simply describe the grammar, but instead the properties of the sentences in the language.

Question 3–4: (Solution, p 6) Consider the following Ada code, based on the parser of Project 1.

```
procedure Parse_Term is
begin
    if Peek_Token = Ident then
        Chomp_Token(Ident);
    else
        Chomp_Token(Num);
    end if;
end Parse_Term;

procedure Parse_Expr is
begin
    if Peek_Token = Semi then
        Chomp_Token(Semi);
    else
        Parse_Term;
        Chomp_Token(Plus);
        Parse_Expr;
    end if;
end Parse_Expr;
```

Give an BNF grammar that describes the language that `Parse_Expr` accepts. The terminals of the language are `PLUS`, `SEMI`, `IDENT`, and `NUM`.

Question 3–5: (Solution, p 6) Consider the following Ada code, based on the parser of Project 1.

```
procedure Parse_S is
begin
    case Peek_Token is
    when A =>
        Chomp_Token(A);
        while Peek_Token = A loop
            Chomp_Token(A);
        end loop;
        Parse_S;
        Chomp_Token(B);
        while Peek_Token = B loop
            Chomp_Token(B);
        end loop;
        Parse_S;
    when B =>
        Chomp_Token(B);
        Parse_S;
        Chomp_Token(A);
        Parse_S;
    when C =>
        Chomp_Token(C);
    when others =>
        Put_Line("unexpected token: expected A, B, or C");
        raise Parse_Error;
    end case;
end Parse;
```

Give an EBNF grammar describing what language this procedure accepts. Your grammar should have only one nonterminal (name it **S**) and three terminals, `A`, `B`, and `C`.

4 *Questions*

Question 4–1: (Solution, p 7) Prove the following using our axiomatic scheme. The notation $\mathit{fib}(i)$ is used to represent the ith Fibonacci number.

$\{a = \mathit{fib}(i)$ **and** $b = \mathit{fib}(i-1)\}$
```
c := a + b;
```
$\{c = \mathit{fib}(i+1)$ **and** $a = \mathit{fib}(i)\}$

Question 4–2: (Solution, p 7) Provide a step-by-step partial correctness proof for the following. Justify each step with one of the three rules below, or "mathematical fact" if the conclusion is algebraically true and involves no assertions.

$\{z = x^i\}$
```
z := z * x;
i := i + 1;
```
$\{z = x^i\}$

Question 4–3: (Solution, p 8) Prove the following using our axiomatic scheme.

$\{i \cdot k = n$ **and** $k \bmod 2 = 0\}$
```
k := k / 2;
i := i * 2;
```
$\{i \cdot k = n;\}$

Question 4–4: (Solution, p 8) Prove the following using our axiomatic scheme

{ **true** }
```
if n mod 2 = 0 then
    n := n - 2;
else
    n := n - 1;
end if;
```
$\{n \bmod 2 = 0\}$

Solution 2–1: (Question, p 1)

```
with Text_IO; use Text_IO;
procedure Print_Large is
    package Int_IO is new Integer_IO(Integer); use Int_IO;
    Nums : array (1..30) of Integer;
begin
    for I in Nums'Range loop
        Get(Nums(I));
    end loop;
    for I in Nums'Range loop
        if Nums(I) > 50 then
            Put(Nums(I)); New_Line;
        end if;
    end loop;
end Print_Large;
```

Solution 2–2: (Question, p 1)

a. `6 8` then `6 8`

b. `5 11` then `11 11` (or `5 11`)

c. `6 12` then `12 12`

Solution 2–3: (Question, p 1)

a. `7 3`

b. `7 2`

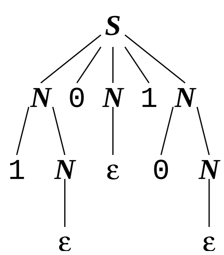
c. `5 2`

Solution 3–1: (Question, p 2)

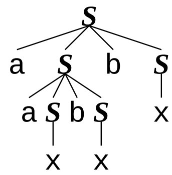
a.

$$\begin{aligned} \mathbf{S} &::= \mathbf{N}\,0\,\mathbf{N}\,1\,\mathbf{N} \\ &::= 0\,\mathbf{N}\,1\,\mathbf{N} \\ &::= 0\,1\,\mathbf{N} \\ &::= 0\,1\,0\,\mathbf{N} \\ &::= 0\,1\,0 \end{aligned}$$

b.



Solution 3–2: (Question, p 2)



Solution 3–3: (Question, p 2)

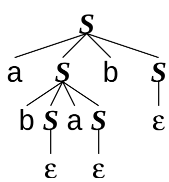
a.

$$\begin{aligned}\mathbf{S} &\Rightarrow \mathtt{b}\,\mathbf{S}\,\mathtt{a}\,\mathbf{S}\\ &\Rightarrow \mathtt{b}\,\mathtt{a}\,\mathbf{S}\\ &\Rightarrow \mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathbf{S}\\ &\Rightarrow \mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathtt{b}\,\mathbf{S}\\ &\Rightarrow \mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathtt{b}\end{aligned}$$

b.

$$\begin{aligned}\mathbf{S} &\Rightarrow \mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathbf{S}\\ &\Rightarrow \mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathbf{S}\\ &\Rightarrow \mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathtt{a}\,\mathbf{S}\,\mathtt{b}\\ &\Rightarrow \mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathbf{S}\,\mathtt{b}\\ &\Rightarrow \mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathtt{b}\\ &\Rightarrow \mathtt{a}\,\mathbf{S}\,\mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathtt{b}\,\mathtt{b}\\ &\Rightarrow \mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathtt{b}\,\mathtt{b}\end{aligned}$$

c.



d. The language is the set of sentences of *a*'s and *b*'s containing an equal number of each.

Solution 3–4: (Question, p 3)

$$\begin{aligned}\textbf{Expr} &::= \mathtt{SEMI} \mid \textbf{Term}\ \mathtt{PLUS}\ \textbf{Expr}\\ \textbf{Term} &::= \mathtt{IDENT} \mid \mathtt{NUM}\end{aligned}$$

Solution 3–5: (Question, p 3)

$$\mathbf{S} ::= \mathtt{A}\,\{\,\mathtt{A}\,\}\,\mathbf{S}\,\mathtt{B}\,\{\,\mathtt{B}\,\}\,\mathbf{S} \mid \mathtt{B}\,\mathbf{S}\,\mathtt{A}\,\mathbf{S} \mid \mathtt{C}$$

Solution 4–1: (Question, p 4)

1.	$a = \mathit{fib}(i)$ **and** $b = \mathit{fib}(i-1)$ **implies** $a + b = \mathit{fib}(i+1)$ **and** $a = \mathit{fib}(i)$	Mathematical fact
2.	$\{a = \mathit{fib}(i)$ **and** $b = \mathit{fib}(i-1)\}$ $\{a + b = \mathit{fib}(i+1)$ **and** $a = \mathit{fib}(i)\}$	Consequence: 1
3.	$\{a + b = \mathit{fib}(i+1)$ **and** $a = \mathit{fib}(i)\}$ `c := a + b;` $\{c = \mathit{fib}(i+1)$ **and** $a = \mathit{fib}(i)\}$	Assignment
4.	$\{a = \mathit{fib}(i)$ **and** $b = \mathit{fib}(i-1)\}$ `c := a + b;` $\{c = \mathit{fib}(i+1)$ **and** $a = \mathit{fib}(i)\}$	Sequence: 2, 3

Solution 4–2: (Question, p 4)

1.	$z = x^i$ **implies** $z \cdot x = x^{i+1}$	Mathematical fact
2.	$\{z = x^i\}$ $\{z \cdot x = x^{i+1}\}$	Consequence: 1
3.	$\{z \cdot x = x^{i+1}\}$ `z := z * x;` $\{z = x^{i+1}\}$	Assignment
4.	$\{z = x^i\}$ `z := z * x;` $\{z = x^{i+1}\}$	Sequence: 3, 4
5.	$\{z = x^{i+1}\}$ `i := i + 1;` $\{z = x^i\}$	Assignment

6.	$\{z = x^i\}$
	`z := z * x;`
	`i := i + 1;`
	$\{z = x^i\}$
		Sequence: 4, 5

Solution 4–3: (Question, p 4)

1. $i \cdot k = n$ **and** $k \bmod 2 = 0$ **implies** $(i \cdot 2) \cdot (k/2) = n$ — Mathematical fact
2. $\{i \cdot k = n$ **and** $k \bmod 2 = 0\}$ — Consequence: 1
   $\{(i \cdot 2) \cdot (k/2) = n\}$
3. $\{(i \cdot 2) \cdot (k/2) = n\}$ — Assignment
   ```
   k := k / 2;
   ```
   $\{(i \cdot 2) \cdot k = n\}$
4. $\{i \cdot k = n$ **and** $k \bmod 2 = 0\}$ — Sequence: 2, 3
   ```
   k := k / 2;
   ```
   $\{(i \cdot 2) \cdot k = n\}$
5. $\{(i \cdot 2) \cdot k = n\}$ — Assignment
   ```
   i := i * 2;
   ```
   $\{i \cdot k = n\}$
6. $\{i \cdot k = n$ **and** $k \bmod 2 = 0\}$ — Sequence: 4, 5
   ```
   k := k / 2;
   i := i * 2;
   ```
   $\{i \cdot k = n\}$

Solution 4–4: (Question, p 4)

1. **true and** $n \bmod 2 = 0$ **implies** $(n-2) \bmod 2 = 0$ — Mathematical fact
2. $\{$ **true and** $n \bmod 2 = 0\}$ — Consequence: 1
   $\{(n-2) \bmod 2 = 0\}$
3. $\{(n-2) \bmod 2 = 0\}$ — Assignment
   ```
   n := n - 2;
   ```
   $\{n \bmod 2 = 0\}$
4. $\{$ **true and** $n \bmod 2 = 0\}$ — Sequence: 2, 3
   ```
   n := n - 2;
   ```
   $\{n \bmod 2 = 0\}$
5. **true and** $n \bmod 2 \neq 0$ **implies** $(n-1) \bmod 2 = 0$ — Mathematical fact
6. $\{$ **true and** $n \bmod 2 \neq 0\}$ — Consequence: 5
   $\{(n-1) \bmod 2 = 0\}$
7. $\{(n-1) \bmod 2 = 0\}$ — Assignment
   ```
   n := n - 1;
   ```
   $\{n \bmod 2 = 0\}$
8. $\{$ **true and** $n \bmod 2 \neq 0\}$ — Sequence: 6, 7
   ```
   n := n - 1;
   ```
   $\{n \bmod 2 = 0\}$
9. $\{$ **true** $\}$ — Selection: 4, 8
   ```
   if n mod 2 = 0 then
       n := n - 2;
   else
       n := n - 1;
   end if;
   ```
   $\{n \bmod 2 = 0\}$