CSCE 222 (Carlisle), Discrete Structures for Computing
Spring 2020
Homework 8

Type your name below the pledge to sign
On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work.
ARTHUR CHEN

Instructions:

- The exercises are from the textbook. You are encouraged to work extra problems to aid in your learning; remember, the solutions to the odd-numbered problems are in the back of the book.
- Grading will be based on correctness, clarity, and whether your solution is of the appropriate length.
- Always justify your answers.
- Don't forget to acknowledge all sources of assistance in the section below, and write up your solutions on your own.
- *Turn in .pdf file to Gradescope by the start of class on Tuesday, March 17, 2020.* It is simpler to put each problem on its own page using the LaTeX clearpage command.

Help Received:

- List any help received here, or "NONE".
  NONE

Exercises for Section 5.3:

6(a,c,d): (3 points). Do NOT do proofs. For 6d it should read $n \geq 2$

a.
$f(0) = 1$
$f(1) = -f(0) = -1$
$f(2) = -f(1) = 1$
$f(3) = -f(2) = -1$
$f(4) = -f(3) = 1$
so we can conclude that

$$f(n) = \begin{cases} \text{-1} & \text{n is odd} \\ 1 & \text{n is even} \end{cases}$$

c.
$f(0) = 0$
$f(1) = 1$
$f(2) = 2 * f(3)$
$f(3) = 2 * f(4)$
since we cannot determine $f(2)$, $f(n)$ is not defined

d.
$f(0) = 0$
$f(1) = 1$
$f(2) = 2 * f(1) = 2$
$f(3) = 2 * f(2) = 4$
$f(4) = 2 * f(3) = 8$
so we can define that

$$f(n) = \begin{cases} 0 & \text{n = 0} \\ 1 & \text{n = 1} \\ 2^{n-1} & n \geq 2 \end{cases}$$

12: (2 points). Proof:
given that f is the series of Fibonacci numbers.
Basic step: $f_1 * f_2 = (1)(1) = 1 = 1^2 = f_1^2$
Inductive step:
Let $f(k)$ be true, $f_1^2 + f_2^2 + ... + f_k^2 = f_k * f_{k+1}$
To prove that $f(k+1)$ is also true
$f_1^2 + f_2^2 + ... + {f_{k+1}}^2$
$= f_1^2 + f_2^2 + ... + f_k^2 + f_{k+1}^2$
$= f_k * f_{k+1} + f_{k+1}^2$
$= f_k * f_{k+1} + f_{k+1} * f_{k+1}$
$= f_{k+1} * (f_k + f_{k+1})$
$= f_{k+1} * f_{k+2}$
since $f(k+1)$ is true, given statement is true for all positive integers n.

18: (3 points).

let $P(n)$ be $\begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$

base step:

$P(1) = A^1 = \begin{bmatrix} f_2 & f_1 \\ f_1 & f_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ = A

which is true

Inductive step:

Assume $P(k)$ is true:

$P(k) = \begin{bmatrix} f_{k+1} & f_k \\ f_k & f_{k-1} \end{bmatrix}$

now proving $P(k+1)$ is also true:

$A^{k+1} = A * A^k$

$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f_{k+1} & f_k \\ f_k & f_{k-1} \end{bmatrix}$

$= \begin{bmatrix} 1*f_{k+1} + 1 * f_k & 1 * f_k + 1 * f_{k-1} \\ 1 * f_{k+1} + 0 * f_k & 1 * f_k + 0 * f_{k-1} \end{bmatrix}$

$= \begin{bmatrix} f_{k+2} & f_{k+1} \\ f_{k+1} & f_k \end{bmatrix}$

so P(k+1) is true.

46: (2 points).
Proving $l(T) = i(T) + 1$:
Basic step:
When the height of the tree T is 0,
$l(T) = 1$
$i(T) = 0$
$l(T) = 0 + 1 = i(T) + 1$
True
Inductive step:
Assume it is also true for the trees T_1 and T_2, which are the children of tree T. So we have:
$l(T) = l(T_1) + l(T_2)$
$i(T) = i(T_1) + i(T_2) + 1$
$l(T_1) = i(T_1) + 1$
$l(T_2) = i(T_2) + 1$
Furthermore,
$l(T) = l(T_1) + l(T_2)$
$= i(T_1) + 1 + i(T_2) + 1$
$= (i(T_1) + i(T_2) + 1) + 1$
$= i(T) + 1$
So statement is true for all tree.

Exercises for Section 5.4:

In the following problems, assume that the following functions are defined:

- def isNull(T : Tree) $\rightarrow$ bool:
- def value(T : Tree) $\rightarrow$ int:# may not be called on Null tree
- def left(T : Tree) $\rightarrow$ Tree: # may not be called on Null tree
- def right(T : Tree) $\rightarrow$ Tree: # may not be called on Null tree

Also, assume a binary search tree. That is, given Tree x, all values in the left subtree are $\leq$ $value(x)$ and all values in the right subtree are $> value(x)$.

Custom 1: (3 points) Write a recursive method that finds the smallest item in the tree.

```
int Min(struct node* node){
    if (isNull(left(node))){
        return value(node);
    }
    return Min(left(node));
}
```

Custom 2: (3 points) Write a recursive method that finds the sum of all the values in the tree.

```
int Sum(struct node* node){
    if (isNull((node)){
        return 0;
    }
    return (value(node) + Sum(left(node)) + Sum(right(node)));
}
```

Custom 3: (2 points) Write a recursive method that returns $True$ if its integer parameter, x, is in the tree and $False$ otherwise.

```
bool find(struct node* node1, struct node* node2, int x){
    if (value(node1) == x){
        return true;
    } else if (value(node2) == x) {
        return true;
    } else if (node1 == nullptr && node 2 == nullptr) {
        return false;
    }
    return find(left(node), right(node), x);
}
```

50: (2 points)
quick sort :
the first number of list is chosen to be pivot.
then divide the list by comparing with the pivot.
first sublist is all the numbers smaller than pivot, and second one is all the numbers that are larger.
recursively repeat this process till every list only contains one number
note that new pivot is chosen every time
solution:
Q(3,5,7,8,1,9,2,4,6)
= Q(1,2,3),Q(5,7,8,9,4,6)
= Q(1),Q(2,3),Q(4,5),Q(7,8,9,6)
= Q(1),Q(2),Q(3),Q(4),Q(5),Q(6,7),Q(8,9)
= Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8),Q(9)
now the list is sorted