Xi Liu, xl3504, Homework 5

Problem 1

```
/* f() is an algorithm that finds the
nth fibonacci number with O(1) extra space
base cases: f(0) = 0; f(1) = 1 */

int f(int n)
{
    int left = 0, right = 1;
    for(int i = 1; i <= n; i++)
    {
        int t = right;
        right = left + right;
        left = t;
    }
    return left;
}
```

Problem 2
(a)

$$\begin{aligned} maxprice(n) = \max(&P[1] + maxprice(n-1) - 1, \\ &P[2] + maxprice(n-2) - 1, \\ &..., \\ &P[n] + maxprice(0)) \end{aligned}$$

$$= \max\Bigg(\max_{1 \le i \le n-1} (P[i] + maxprice(n-i) - 1),$$
$$P[n] + maxprice(0)\Bigg)$$

the last argument, $P[n] + maxprice(0)$, in max() function corresponds to having zero cuts and directly selling the rod of length n, since there is no cuts, there is no cost of cutting for the last argument. $\max_{1 \le i \le n-1}(P[i] + maxprice(n-i)-1)$ are the other $n-1$ arguments in max() that correspond to the maximum price obtained by making a cut of the rod into a left-hand piece of length i and a right-hand remainder of length $n-i$, $\forall i \in [1, n-1] \cap \mathbb{N}$, and only the right-hand remainder can be cut further
all of the $n-1$ arguments in $\max_{1 \le i \le n-1}(P[i]+maxprice(n-i)-1)$ involves an extra cut and a corresponding cut cost of $1 in addition to the cuts already made in the $maxprice(n-i)$ part

(b)
base case:

$$maxprice(0) = 0$$

/* when the rod has a length of 0 inch,
the price is \$0, and there is no cut */

$$maxprice(1) = P[1]$$

/* when the rod has a length of 1 inch,
the price is P[1], and there is no cut */

(c)

```
/* maxprice() returns the maximum selling price
we can get among all possible options,
with a payment of $1 per cut */

int max(int a, int b)
{  return (a > b) ? a : b;  }

int maxprice(int * P, int n)
{
    int memo[n + 1];
    *memo = 0;
    for(int i = 1; i <= n; i++)
    {
        memo[i] = P[i];
        for(int j = 1; j <= i; j++)
        {
            memo[i] = max(memo[i], P[j] + memo[i - j] - 1);
        }
    }
    return memo[n];
}
```

let $T(n)$ be the running time of maxprice()

$$T(n) = \Theta(n^2)$$

since there are 2 levels of nested for loops and the line within the innermost level take constant time
the line within the innermost level take constant time since values at $memo[i]$ and $P[i]$ are readily available to be fetched and max() function only involves a constant time comparison

$$\begin{aligned}
T(n) &= \text{(number of loop iterations)(cost of innermost level)} \\
&= \text{(number of loop iterations)} \\
&= \sum_{i=1}^{n} i \\
&/\text{* arithmetic series with } a_1 = 1, a_n = n \text{ */} \\
&= \frac{n(a_1 + a_n)}{2} \\
&= \frac{n(1+n)}{2} \\
&= \frac{n}{2} + \frac{n^2}{2} \\
&= \frac{n^2}{2} + \frac{n}{2} \\
&= \Theta(n^2)
\end{aligned}$$

Problem 3

(a)

$$
\begin{aligned}
canreach(n) &= canreach(n-1) \vee canreach(n-2) \vee ... \vee canreach(n-A[n]) \\
&= \bigvee_{i=1}^{A[n]} canreach(n-i)
\end{aligned}
$$

/* $\vee$ is or */

(b)

$$canreach(1) = true$$

/* when the frog is at $A[1]$, the frog reaches the destination (so the frog can reach the 1st index of the array), the function returns true */

(c)

```c
#include <stdbool.h>

bool canreach(int * A, int n)
{
    bool memo[n + 1];
    memo[1] = true;
    for(int i = 1; i <= n; i++)
    {
        if(A[i] == 0)
        {
            memo[i] = false;
            continue;
        }
        else if(i - A[i] <= 1)
        {
            memo[i] = true;
            continue;
        }
        else
        {
            for(int j = 1; j <= A[i] && i - j >= 1; j++)
            {
                if(memo[i - j] == true)
                {
                    memo[i] = memo[i - j];
                    continue;
                }
            }
        }
    }
    return memo[n];
}
```

let $T(n)$ be the running time of canreach() in the worst case

$$T(n) = O(n^2)$$

since there are 2 levels of nested for loops and the lines within the innermost level take constant time
the lines within the innermost level take constant time since values at $memo[i-j]$ and $A[i]$ are readily available to be fetched
in the worst case, $A[i] = i - 1$

$$\begin{aligned}
T(n) &= \text{(number of loop iterations)(cost of innermost level)} \\
&= \text{(number of loop iterations)} \\
&= \sum_{i=1}^{n} A[i] \\
&= \sum_{i=1}^{n} (i-1) \\
&= \sum_{i=1}^{n} i - \sum_{i=1}^{n} 1 \\
&= -(n-1+1)1 + \sum_{i=1}^{n} i \\
&= -n + \sum_{i=1}^{n} i \\
&\text{/* arithmetic series with } a_1 = 1, a_n = n \text{ */} \\
&= -n + \frac{n(a_1 + a_n)}{2} \\
&= -n + \frac{n(1+n)}{2} \\
&= -n + \frac{n}{2} + \frac{n^2}{2} \\
&= \frac{n^2}{2} - \frac{n}{2} \\
&= O(n^2)
\end{aligned}$$

Problem 4

(a)

$$makechange(8) = 5$$

$i = 8$: There are 5 ways:
$1 + 1 + 1 + 1 + 1 + 1 + 1 + 1$ (use 8 pennies),
$1 + 1 + 1 + 5$ (3 pennies and 1 nickel),
$1 + 1 + 5 + 1$ (2 pennies, 1 nickel, and 1 penny),
$1 + 5 + 1 + 1$ (1 penny, 1 nickel, and 2 pennies),
$5 + 1 + 1 + 1$ (1 nickel and 3 pennies)

$$makechange(9) = 6$$

$i = 9$: There are 6 ways:
$1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1$ (use 9 pennies),
$1 + 1 + 1 + 1 + 5$ (4 pennies and 1 nickel),
$1 + 1 + 1 + 5 + 1$ (3 pennies, 1 nickel, and 1 penny),
$1 + 1 + 5 + 1 + 1$ (2 pennies, 1 nickel, and 2 pennies),
$1 + 5 + 1 + 1 + 1$ (1 penny, 1 nickel and 3 pennies)
$5 + 1 + 1 + 1 + 1$ (1 nickel and 4 pennies)

(b)

$$\begin{aligned}
&\forall n \geq 10,\ n \in \mathbb{N} \\
&makechange(n) \\
&= makechange(n-1) + makechange(n-5) + makechange(n-10)
\end{aligned}$$

(c)

```
int makechange(int n)
{
    int memo[n + 1];
    for(int i = 1; i <= n; i++)
    {
        if(i <= 4)
            memo[i] = 1;
        else if(i == 5)
            memo[i] = 2;
        else if(n >= 5)
        {
            memo[i] = memo[i - 1]
                    + memo[i - 5];
        }
        else if(n >= 10)
        {
            memo[i] = memo[i - 1]
                    + memo[i - 5]
                    + memo[i - 10];
        }
    }
    return memo[n];
}
```

let $T(n)$ be the running time of makechange()

$$T(n) = \Theta(n)$$

since there is 1 level of for loop and the lines within the for loop take constant time

the lines within the for loop take constant time since values at $memo[i]$, $memo[i-1]$, $memo[i-5]$, $memo[i-10]$ are readily available to be fetched and the if statements are constant time comparisons

$$
\begin{aligned}
T(n) &= \text{(number of loop iterations)(cost of innermost level)} \\
&= \text{(number of loop iterations)} \\
&= \sum_{i=1}^{n} 1 \\
&= n - 1 + 1 \\
&= n \\
&= \Theta(n)
\end{aligned}
$$