McMaster University
SFWR ENG 2MD3 Winter 2020
Assignment 2
Name: Manuel R. Lemos
Student Number: 400177763
Student ID: lemosm1
Due: Sunday February 2, 2019 at 23:55

Recursion (35 marks)

Question 1 (4 marks)

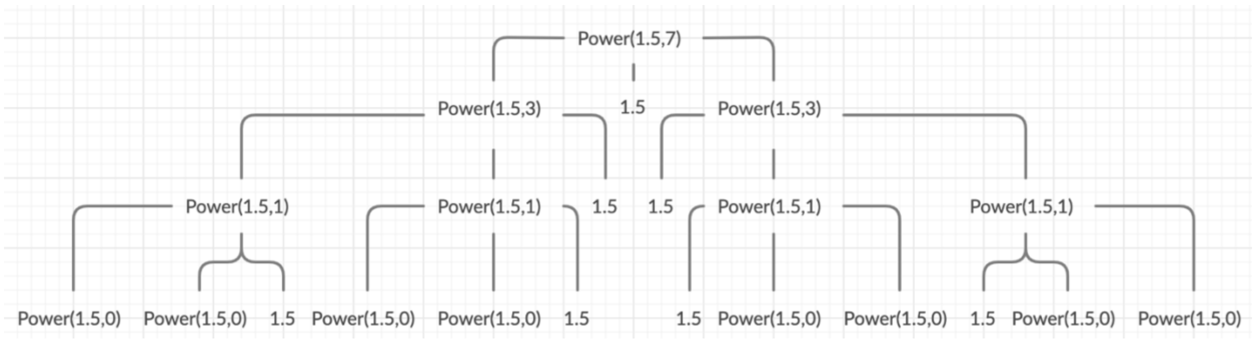
```
float Power(float x, int n){
    if (n == 0){return 1;}
    else {return (x*Power(x,n-1));}
}
```

Question 2 (10 marks)

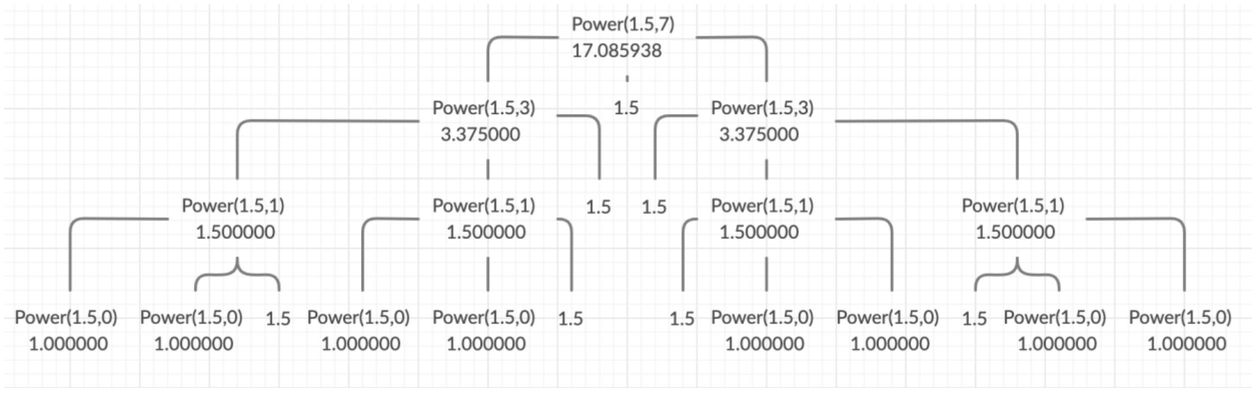
1. Provide answer to question 2 of Exercise 3.2 of the book [4 marks]

```
float Power(float x, int n){
    float accum;
    if(n == 0){return 1;}
    else{
        accum = Power(x,n/2);
        if(!(n%2)){return (accum * accum);}
        else{return (accum * accum * x);}
    }
}
```

2. Provide the call tree for Power (1.5, 7) [2 marks]



3. Provide the annotated call tree for Power (1.5, 7) [2 marks]



4. Provide the call trace for Power (1.5, 7) [2 marks]

Power(1.5,7)
= Power(1.5,3) * Power(1.5,3) * 1.5
= [Power(1.5,1) * Power(1.5,1) * 1.5] * [Power(1.5,1) * Power(1.5,1) * 1.5] * 1.5
= [1.5 * 1 * 1 * 1.5 * 1 * 1 * 1.5] * [1.5 * 1 * 1 * 1.5 * 1 * 1 * 1.5] * 1.5
= [3.375] * [3.375] * 1.5
= 17.085938

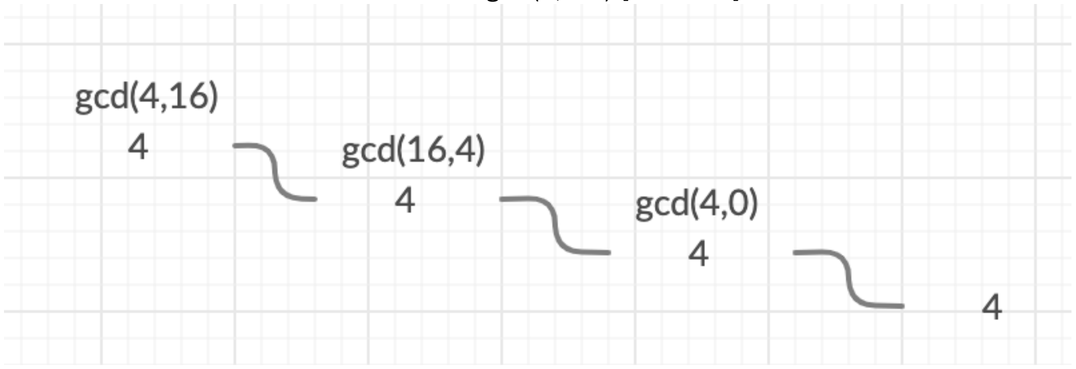
Question 3 (8 marks each part)

1. Provide answer to question 4 of Exercise 3.2 of the book.[4 marks]

```
int gcd(int m, int n)
{
    if (n==0) return m;
    int r = m%n;
    return gcd(n,r);

}
```

2. Provide the annotated call tree for gcd(4, 16) [2 marks]



3. Provide the call trace for gcd(4, 16) [2 marks]

gcd(4,16)
= gcd(16,4)
= gcd(4,0)
= 4

Question 4 (5 marks)

```
char * Head(char * str)
{
    char * head;
    head = malloc(2*sizeof(char));
    head[0] = str[0];
    head[1] = '\0';
    return head;
}

char * Tail(char * str)
{
    return ++str;
}

char * Concat(char * str1, char * str2)
{
    char * sconcat;
    int i = 0;
    int length = strlen(str1) + strlen(str2);
    sconcat = malloc(length*sizeof(char));
    while(i<strlen(str1)){
        sconcat[i] = str1[i];
        i++;
    }
    while(i<length){
        sconcat[i] = str2[i-strlen(str1)];
        i++;
    }
    sconcat[i] = '\0'; //indicates end of string preventing weird errors
    return sconcat;
}

char * Reverse(char * str)
{
    if (strlen(str) == 0) {return "";}
    else {return Concat( Reverse(Tail(str)), Head(str) );}
}
```

The overall efficiency when compared to the recursive reversal program is poor. Runtime is greater as a result of repeated scanning which is not present in the recursive program. Furthermore, dynamic storage allocations are not freed, compounding program ineffiency.

Question 5 (4 marks)

```
int LinkedListLength(NodeType * L)
{
    if (L == NULL){return 0;}
    else{return (1 + LinkedListLength(L->Link));}
}
```

Question 6 (2 marks)

All digits 0 through 9 are assigned to their respective alphabetical characters **'O'** through **'X'** using the char and integer casting within the **PDigit(int d)** function.

If the integer entered is 9 or less:

The function passes n to **PDigit(int d)** which then assigns n to its respective letter and prints that letter out

If the integer, n, entered is 10 or greater:

The function calls itself with the last digit removed

This will continue until only a single digit is present and the corresponding character Is printed

Once the first digit's letter has been printed, the function will begin cycling out of recursion layers and begins printing the last digit from each of the n values passed

The summation of these steps results in the letter corresponding to each number being printed out in the same order the values were entered in**. ie) 1234 -> PQRS**

Question 7 (2 marks)

The function **R(int n)** prints out the reverse of the number entered

If the integer entered is 9 or less:

The function prints n%10 which given a single digit will print that digit

If the integer, n, entered is 10 or greater:

The function prints the last digit of the integer using n%10

Then the function checks if n>9. If so it recurses passing R one tenth of n, **R(n/10)**, effectively removing the last digit from the integer provided and beginning the process again.