Computer Science I
Exercise: Recursion

We have gone through many examples in the class and those examples are available in the slides and uploaded codes. Try to test those codes and modify as you wish for getting more clarification.

In addition try the following:

1) What would be the output of the following recursive function if we call rec2(5) ?

```
void rec2(int x)
{
    if (x==0)
        return;
     rec2(x-1);
    printf("%d  ", x);
}
```

Answer: 1 2 3 4 5

2) Write a recursive function that calculates the sum $1^1 + 2^2 + 3^3 + \ldots + n^n$, given an integer value of n in between 1 and 9. You can write a separate power function in this process and call that power function as needed:

```
int crazySum(int n){
    if (n == 0)
    { return 0;
    }
    else
    { return  power(n,n) + crazySum(n-1);
    }
}
```

```
int  power(int a, int b) {
    if (b == 0)
    { return 1;
    }
    else
    { return  a * power(a,b-1);
    }
}
```

3) Given the function below, what would the function call question3(10, 101) return?

```
int question3(int a, int b) {
        if (a == 0) return b;
        if (b == 0) return a;
        return question3(10*a+b%10, b/10);
}
```

Answer: 10101

4) Write a recursive function that converts a decimal number to an octal number.

```
int convertDecToOct(int n) {
    if(n==0) {
      return 0; // could return n instead...
    }
    else {
      return (n%8) + 10 * convertDecToOct(n/8);
    }
```

5) The code below returns the number of zeros at the end of n! [factorial n]

```
int zeros(int n)
{
        int res = 0;
        while (n!=0)
        {
                res  += n/5;
                n /= 5;
        }
        return res;
}
```

Rewrite this method recursively:

```
int zeros(int n) {
    if(n==0){
        return 0;
    }
    else {
        return (n/5)+ zeros(n/5);
    }
}
```

6. Write a recursive function that returns the product of the digits of its integer input parameter, n. You may assume that n is non-negative. For example, productDigits(243) should return 24, since 2 x 4 x 3 = 24.

```
int productDigits (int n) {
    if(n==0){
        return 1;
    }
    else {
        return (n%10) * productDigits((n-n%10)/10);
    }
}
```

7. Let us define the weighted sum of an integer array a[0], a[1], a[2], ..., a[n-1] be a[0]*1 + a[1]*2 + a[2]*3 + ...+a[n-1]*n. For example, the weighted sum of the array [5,2,6] would be 5*1+2*2+6*3 = 27. Write a recursive function that takes in an array numbers and its length n, and returns its weighter sum. You can assume n is non-negative integer.

```
int weightedSum(int numbers[], int n) {
    if(n==0){
        return 0;
    }
    else {
        return (numbers[n-1] * n) + weightedSum(numbers, n-1);
    }
}
```

8. Mathematically, given a function f, we recursively define $f^k(n)$ as follows: if $k = 1$, $f^1(n) = f(n)$. Otherwise, for $k > 1$, $f^k(n) = f(f^{k-1}(n))$. Assume that there is an existing function f, which takes in a single integer and returns an integer. Write a ***recursive*** function fcomp, which takes in both n and k ($k > 0$), and returns $f^k(n)$.

```
int f(int n);
int fcomp(int n, int k){
    if(k==1){
        return f(n);
    }
    else if(k>1){
        return fcomp(n,k-1) * f(n);
    }
    else {
        printf("ERROR - number cannot be negative \n");
        return 0;
    }
}
```

9. What would be the value of fun(7) for the following function?

```
int fun(int x)
{
  if(x==0)
    return 0;
  if(x%3 ==0)
    return fun(x/3);
  return fun(x-1) + x;
}
```

fun(7) = 10

// 0+1=1...
// 1+2=3...
// 3+7=10...

10. Draw the recursion tree to find out the value of f(5)

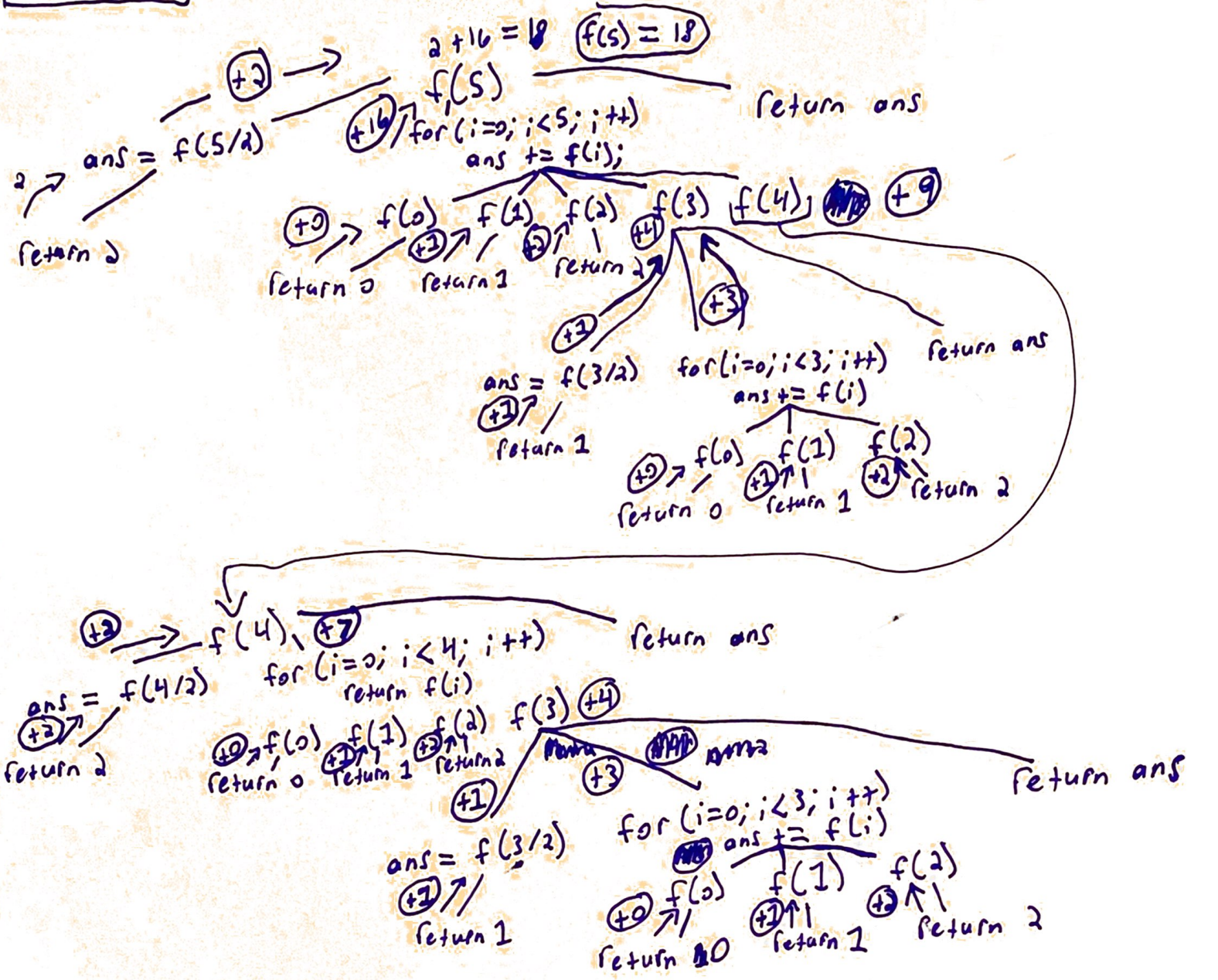
```
int f(int n)
{
        int ans;
        int i;
        if(n<3)
                return n;
        ans = f(n/2);
        for(i=0; i<n; i++)
                ans += f(i);
        return ans;
}
```

recursion tree for f(5) on back ...

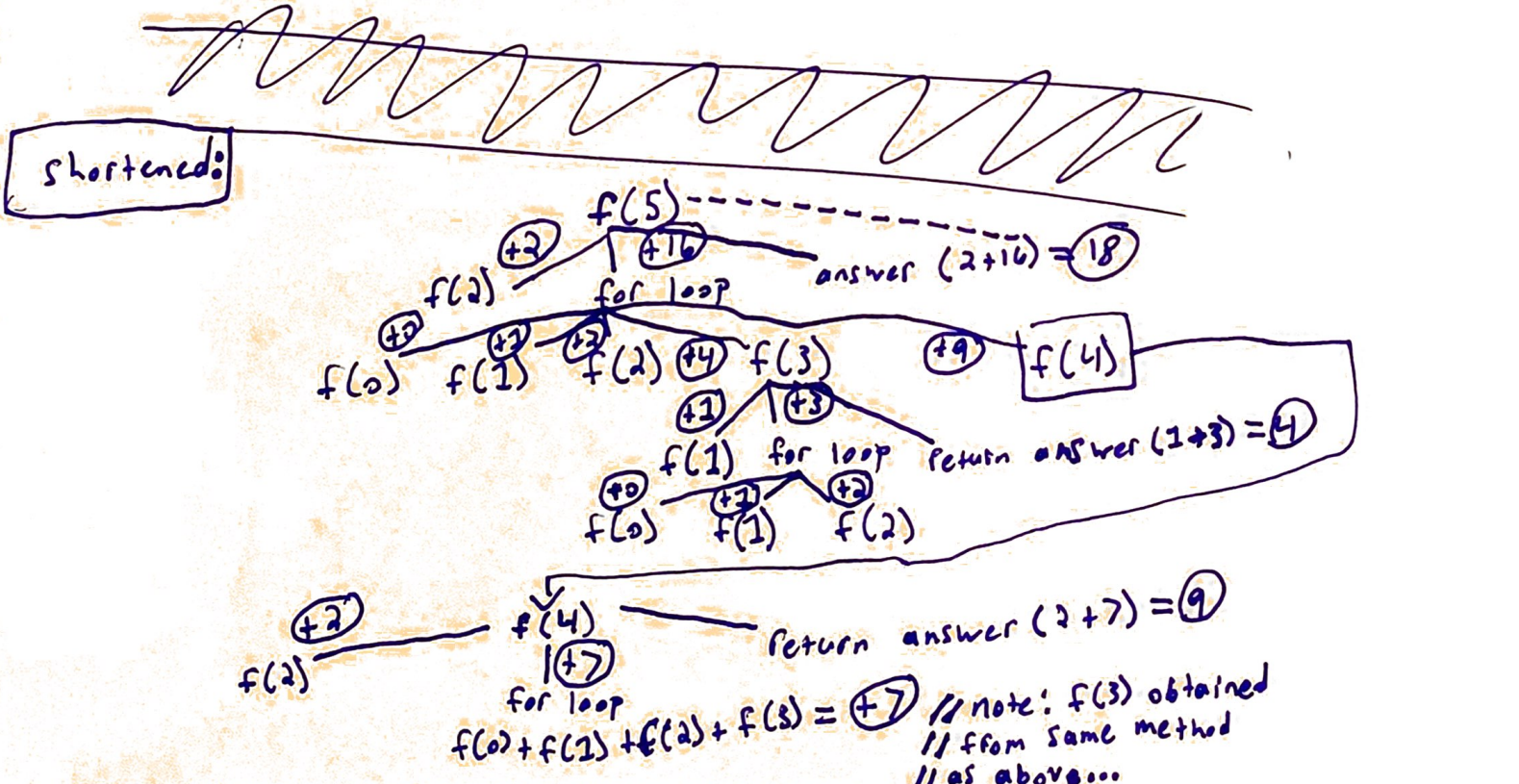
11. Write a recursive function that returns 1 if an array of size n is in sorted order and 0 otherwise. Note: If array a stores 3, 6, 7, 7, 12, then isSorted(a, 5) should return 1 . If array b stores 3, 4, 9, 8, then isSorted(b, 4) should return 0.

```
int isSorted(int *array, int n){
    if(n>1){
        if(array[n-1] >= array[n-2]){
            return 1 * isSorted(array, n-1);
        }
        else {
            return 0;
        }
    }
    else{
        return 1;
    }
}
```

Complete:



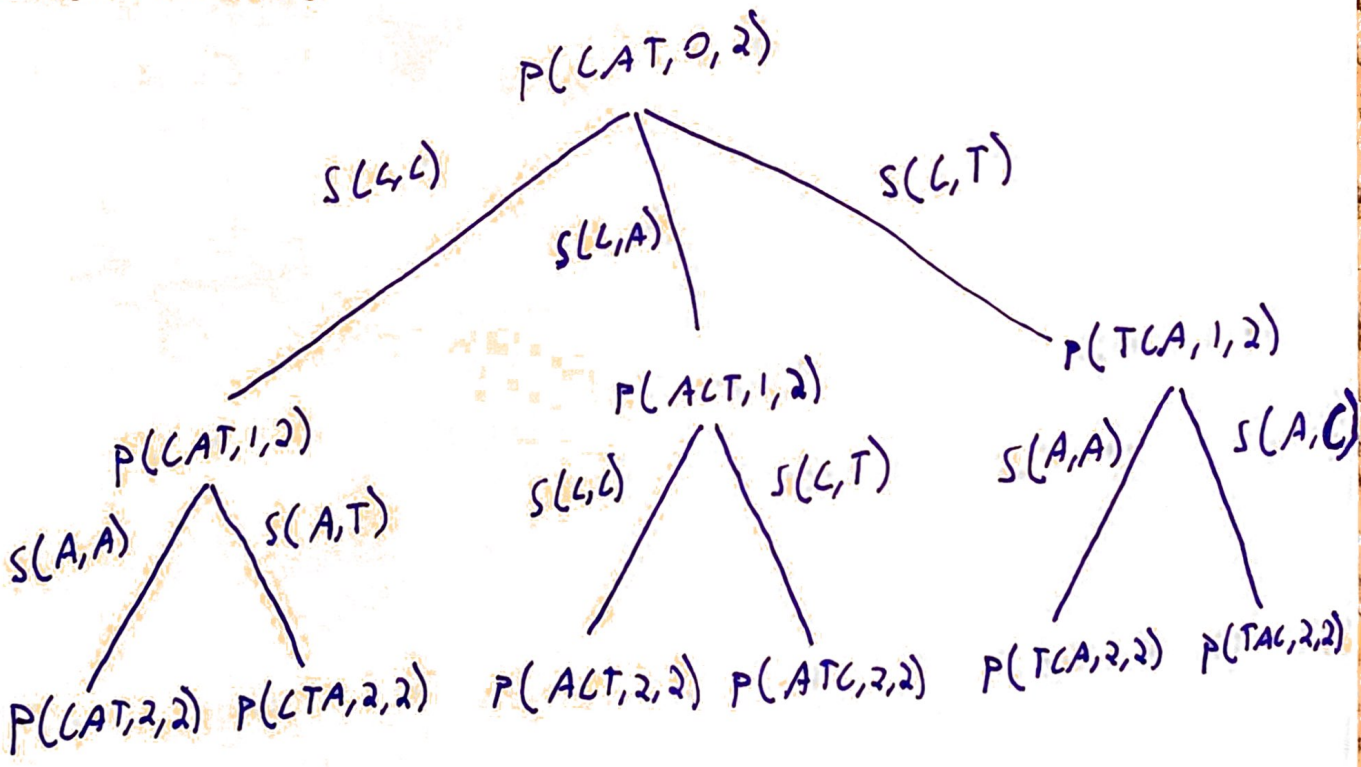
Shortened:



12. There is a recording on permutation in webcourses. In that recording the following code is discussed.

```
void permute(char *s, int l, int r)
{
   int i;
   if (l == r)
     printf("%s\n", s);
   else
   {
       for (i = l; i <= r; i++)
       {
          swap(&s[l], &s[i]);
          permute(s, l+1, r);
          swap(&s[l], &s[i]); //swap
       }
   }
}
```

Draw the recursion tree and mark all the printed strings in the recursion tree after calling the function *permute("CAT", 0, 2). Follow the approach we have learned during the lecture. While drawing, you can write the function call in short form like swap->S(), Permute -> P() to reduce the space required while drawing.*



13. Write an ***efficient recursive*** function that takes in a ***sorted*** array of numbers, two integers, low and high, representing indexes into the array, and another integer, value, and returns the index in the array where value is found in the array in between index low and high, inclusive. If value is NOT found in the array in between indexes low and high, inclusive, then the function should return -1.

```
int search(int numbers[], int low, int high, int value){
    int mid = 0;
    if(low <= high){
        mid = (low + high)/2;
        if(value < numbers[mid])
            return search(numbers, low, mid-1, value);
        else if(value > numbers[mid])
            return search(numbers, low, mid+1, value);
        else
            return mid;
    }
    else
        return -1;
}
```

14. Write recursive functions for inserting an item to the end of a linked list. Also, write another recursive function to delete a specific items from a linked list. ***// you don't have to submit this answer. But it would be good practice, after leaning linked list!***