Announcements

- Lab grades will be uploaded in the next week
- Quiz 1 will be on Friday, 11 AM to 12 PM, Computer Lab 2

Lecture 4
Recursion

Calculate Factorial!

$$f(n) = 1.2.3...(n-1).n$$

```
int fact(int n){
    int cfact = 1;
    for(i=n; i--, i>0)
        cfact=cfact*i;
    return cfact;
}
```

Recursive definition

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot f(n-1) & else \end{cases}$$

```
int fact(int n){
    if(n==1)
        return 1;
    return n*fact(n-1);
}
```

Recursion

- self similar repetition
- function calls itself
- base case or termination condition

```
int fact(int n){
    if(n==1)
        return 1;
    return n*fact(n-1);
}
```

Recursion Vs Iteration

```
int fact(int n){
    if(n==1)
        return n;
    return n*fact(n-1);
}
```

```
int fact(int n){
    int cfact = 1;
    for(i=n; i--, i>0)
        cfact=cfact*i;
    return cfact;
}
```

Fibonacci Series

Recursion

```
int fb(int n){
    if(n==0||n==1)
        return n;
    return fb(n-1)+fb(n-2);
}
```

Iteration

```
int fb(int n) {
int x = 0, y = 1, z = 1;
for(int i = 0; i < n; i++){
    x = y; y = z;
    z = x + y; }
return x; }
```

Exercise: can recursion ever be faster than iteration?

Linear Recursion: a function calls exactly once to itself!

Linear Recursion

- Test for base cases
  - Begin with test cases (at least 1)
  - Chain of recursive call must end at base case
- Recur once
  - Perform a single recursive call
  - Can choose one of many
  - Each call must progress towards base case

Sum of first n Array Elements

```
int linearSum(int A[], int n){
if (n == 1)
    return A[0];
return linearSum(A, n - 1) + A[n - 1];
}
```

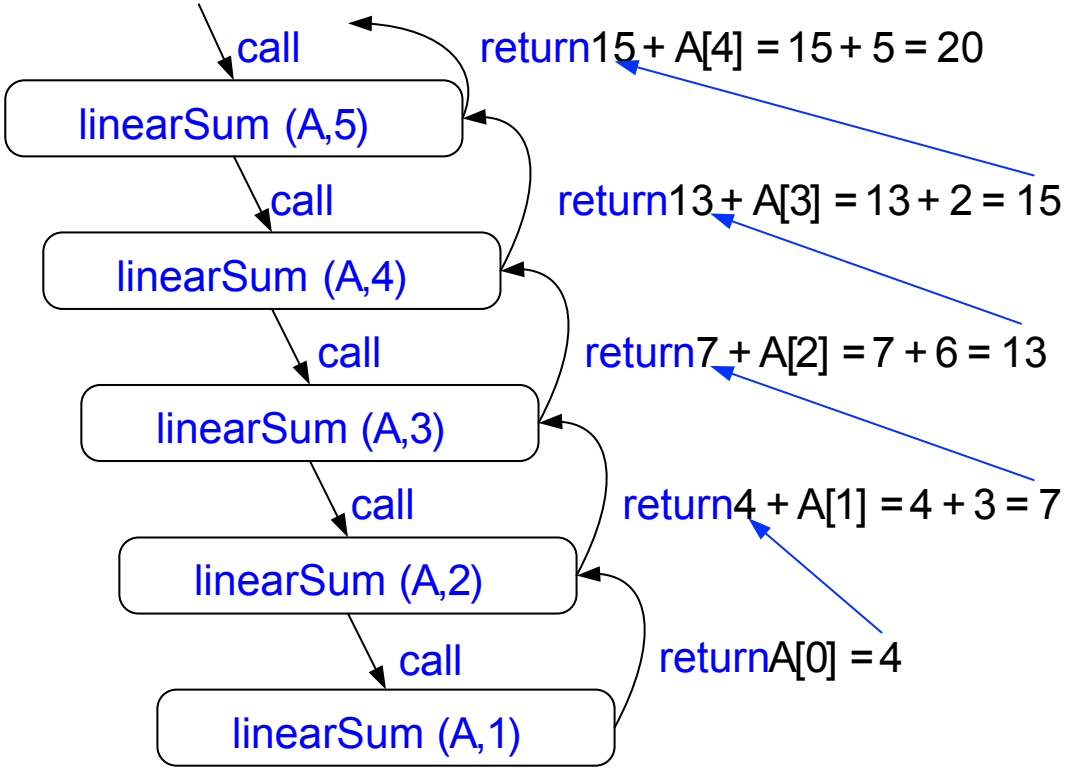
You call:
linearSum(A,n);

Reversing an Array

```
int revArray(int A[], int i,  int j){
    if (i>=j)
        return;
    swap(A(i), A(j));
    return revArray(A, ++i, --j );
}
```

You call:
revArray(A,0,n-1);

Depth of recursion!



Defining Arguments for Recursion

- function definition should facilitate recursion
- sometimes need additional parameters
- E.g. revArray(A,i,n) instead of revArray(A)

Binary Recursion makes two recursive calls each time!

Fibonacci Series

```
int fb(int n){
    if(n==0||n==1)
        return n;
    return fb(n-1)+fb(n-2);
}
```

Find Max in an Array

```
int max(int A[], int I, int j){
    if (i==j)
        return A[i];
    int m = (i+j)/2;
    int u = max(A, i, m);
    int v = max(A, m+1, j);
    if (u>v) return u; else return v;
}
```

Tail Recursion: last step is a recursive call!

No

```
int fb(int n){
     if(n==0||n==1)
          return n;
     return fb(n-1)+fb(n-2);
}
```

Yes

```
int revArray(int A[], int i,  int j){
if (i>=j)
     return;
swap(A(i), A(j));
return revArray(A, ++i, --j );
}
```

Reversing Array

Recursive

```
int revArray(int A[], int i,  int j){
if (i>=j)
      return;
swap(A(i), A(j));
return revArray(A, ++i, --j );
}
```

Iterative

```
int revArray(int A[], int i,  int j){
      while (i<j){
            swap(A(i), A(j));
            i++; j--;
      }
}
```

Fibonacci Series

```
int fb(int n){
    if(n==0||n==1)
        return n;
    return fb(n-1)+fb(n-2);
}
```

Exercise: write tail recursive version!

Multiple Recursion makes more than two recursive calls each time!