CS211
Jerry Waxman

# Examples of simple recursive functions

**Formulating a recursive function:**

**Functions**
We want to solve some computational problem P for some "size" n by designing a function f. Function f may operate by calling other "helper functions" to help it accomplish its task. There is nothing either new or strange about this. We do it all the time. We break up a complex task into different, smaller, more manageable tasks and our function f works by calling the other functions and utilizing their results. For example, f may use certain functions from the math library or other functions that we ourselves have written.

**Recursive functions**
Sometimes, the "helper function" for f may be f itself. If this is the case, them we say that f is defined recursively. The solution to many problems can be naturally thought of in this way. The key to recognizing such problems is to ask ourselves: "could we get the solution for problem p for an "instance of size n" by utilizing the solution for the **same problem** p for an instance of size smaller than n.

It is important to see the difference between use of a helper function in recursion and the ones discussed in the first paragraph above. There we are combining the solution to a number of **different kinds** of sub problems to solve the larger problem. With recursion, the sub problems are of **the same kind** as the large problem, but are just "smaller" instances.

A familiar example is the recursive solution to computing n!. Since

$$n! = n*(n-1)!$$

(in the appropriate range and with the base case defined) we can design a function f to compute n! in terms of the result of **the same function f** computing (n-1)!.

```
int f(int n){

if( n==0) return 1;

        return n*f(n-1);
}
```

What follows below are some simple problems that naturally allow for a similar analysis.

**// recursive power , compute $x^n$**

```
int exp(int x, int n){
        if(n==_______)
                return _________;
        return _____________*exp(_________);
}

void main(){
        int a,b;
        cin >>a>>b;
        cout<<endl<<exp(a,b)<<endl;
}
```

**// recursive print of a string**

```
void print(char* s){
        if(*s==_____________)
                return;
        cout<<*s;
        print(_________________);
}

void main(){
        char a[6]="hello";
        print(a);
}
```

**// recursive reverse print of a string**

```
void rev_print(char* s){
        if(*s==_____________)
                return;
        ___________________________________;

        ___________________________________;
}

void main(){
        char a[100];
        cin>>a;
        rev_print(a);
}
```

**// recursive reverse print of an integer**

```
void rev_print(int n){
        if(n/10==0){
                cout<<n%10;
                return;
        }
        cout<<_____________________________;
                rev_print(_____________________);

}

void main(){
        int a;
        cin>>a;
        rev_print(a);
}
```

**// recursive print of an integer**

```
void print(int n){
                if(n/10==0){
                cout<<n%10;
                return;
        }
        print(_________________);
        cout<<_____________________;

}

void main(){
int a;
cin>>a;
        print(a);
}
```