CSCI 135 Software Design and Analysis
Example functions and programs
we have seen in class

```
//the factorial function
//note the similarity between this
//and the sum of the first n integers
//also note that we initialize a product
//to 1 as opposed to 0 for a sum
//issues: factorial grows fast, might
//want to use something other than int
int fact(int n) {
  int p=1;
  for (int i=1; i<=n; i=i+1)
    p=p*i;
  return p;
}

//the 3n+1 problem
//it is believed that this will
//always reach 1 when you start
//with a positive integer
int collatz(int n) {
  if (n%2==0)
    return n/2;
  else
    return 3*n+1;
}

//note the use of while because we
//don't know explicitly how long
//we will iterate as above (we used for)
int main() {
  int n;
  cout<<"input n: ";
  cin>>n;
  cout<<n;
  while (n!=1) {
    n=collatz(n);
    cout<<"-->"<<n;
  }
  cout<<'\n';
}
```

```
//tricky condition about socks
//the order of checking conditions is
//sometimes important
//for instance, what happens if you check
//for red first? you have to be careful
//how it is done
int main() {
  bool red;
  bool cotton;
  cout<<"Are your socks red? ";
  cin>>red;
  cout<<"Are your socks cotton? ";
  cin>>cotton;

  if (red && cotton)
    cout<<"they are red and cotton";
  else if (red)
    cout<<"they are red";
  else if (cotton)
    cout<<"they are cotton";
  else
    cout<<"they are plain";
  cout<<'\n';
}


//Fibonacci: 0 1 1 2 3 5 8 13...
//keep two variables a and b, and in
//every iteration, a becomes b and b becomes
//a+b, but this cannot be done in one shot,
//so we use a temporary variable.
//issues: fibonacci also grows fast, consider
//using unsigned long int
//always check boundary conditions, e.g. what
//happens if n is zero?
int fib(int n) {
  int a=1;
  int b=0;
  for (int i=1; i<=n; i=i+1) {
    int c=a;
    a=b;
    b=c+b;
  }
  return b;
}
```

```
//actually there is a trick that works here
//without using a temporary, not necessarily
//better for clarity
int fib2(int n) {
  int a=1;
  int b=0;
  for (int i=1; i<=n; i=i+1) {
    b=a+b;
    a=b-a;
  }
  return b;
}

//the greatest common divisor, same idea of
//iteratively updating two variables.
//example: gcd(30,18)=gcd(18,12)=gcd(12,6)=gcd(6,0)=6.
int gcd(int a, int b) {
  while (b!=0) {
    int c=a;
    a=b;
    b=c%b;
  }
  return a;
}

//netsted loops and the pascal triangle.
//1
//1  1
//1  2  1
//1  3  3  1
//1  4  6  4  1
//1  5 10 10  5  1
//...
//if we start with 1 in row n, the kth entry
//can be obtained from the (k-1)st entry by
//multiplying it my (n-k+1)/k, k=1..n
int main() {
  for (int n=0; n<10; n=n+1) {
    int a=1;
    cout<<a;
    for (int k=1; k<=n; k=k+1) {
      a=a*(n-k+1)/k;
      if (a<100)     //for
        cout<<' ';   //a
      if (a<10)      //proper
        cout<<' ';   //alignment
      cout<<' '<<a;
    }
    cout<<'\n';
  }
}
```

```
//generating the divisors of a number n
int main() {
  int n=10; //any value you want
  cout<<"the divisors of "<<n<<": 1"; //starting with 1
  for (int i=2; i<n; i=i+1)
    if (n%i==0)
      cout<<' '<<i;
  cout<<' '<<n<<'\n';
}

//convert above to output divisors of all numbers from 1 to 100
int main() {
  for (int n=1; n<=100; n=n+1) { //add a loop
    cout<<"the divisors of "<<n<<": 1"; //starting with 1
    for (int i=2; i<n; i=i+1)
      if (n%i==0)
        cout<<' '<<i;
    cout<<' '<<n<<'\n';
  }
}

//change above to find all primes from 1 to 100
int main() {
  for (int n=2; n<=100; n=n+1) { //add a loop
    int count=0;
    for (int i=2; i<n; i=i+1)
      if (n%i==0)
        count=count+1;
    if (count==0)
      cout<<n<<'\n';
  }
}

//another way
int main() {
  for (int n=2; n<=100; n=n+1) { //add a loop
    bool prime=true;
    for (int i=2; i<n; i=i+1)
      if (n%i==0)
        prime=false;
    if (prime)
      cout<<n<<'\n';
  }
}
```

```
//a better way
int main() {
  for (int n=2; n<=100; n=n+1) { //add a loop
    bool prime=true;
    for (int i=2; i<n; i=i+1)
      if (n%i==0) {
        prime=false;
        break; //break out of the closest loop, not need to check further
      }
    if (prime)
      cout<<n<<'\n';
  }
}

//one last improvement
int main() {
  for (int n=2; n<=100; n=n+1) { //add a loop
    bool prime=true;
    for (int i=2; i<=sqrt(n); i=i+1)  //no need to check beyong sqrt(n)
      if (n%i==0) {                   //every divisor >=sqrt n corresponds
        prime=false;                  //to a divisor <=sqrt(n)
        break; //break out of the closest loop, not need to check further
      }
    if (prime)
      cout<<n<<'\n';
  }
}

//generate all pairs (a,b) such that 1<=a,b<=100
//and a and b are coprimes, i.e. gcd(a,b)=1
int main() {
  for (int a=1; a<=100; a=a+1)
    for (int b=1; b<=100; b=b+1) //start with b=a+1 to avoid duplicate pairs
      if (gcd(a,b)==1)
        cout<<'('<<a<<','<<b<<")\n";
}
```