Day 18

eat a meal:

```cpp
void eatMeal (int &wtClarkKent) {
    wtClarkKent += 10;
}

int main() {
    int wtSuperman=220;
    eatMeal (wtSuperman);
    cout << wtSuperman << endl;
    return 0;
}
```
Example program

- Generate n random points in a circle
- developing a program from scratch
- idea: keep generating random points, keeping the ones inside the circle

- general algorithm:
  - input n
  - for each random point
    - generate random point in unit circle
    - output point

- example of stubbing
Detailed algorithm

- how to build random point in unit circle
  - build random x coordinate in [-1,1]
  - build random y coordinate in [-1,1]
  - test if inside circle
    - distance from (0,0) <= 1
  - if not, try again
- generate random number in [-1,1]
Day 19

- handout of randCircle.cpp
- lab midterm instructions
  - textbook, bring a book for 8am lab, no excuses for leaving (go to the restroom before), don’t need to comment your code (unless bizarre code)
  - cheating
- solution of HW3: bubble/selection sort
Character arrays

- char name[2]; name[0] = ‘j’; name[1] = ‘j’;
- char word[4] = “cat”; // character string
- equivalent to
  - char word[4]; word[0] = ‘c’; word[1] = ‘a’;
    word[2] = ‘t’; word[3] = ‘\0’;
- more at end of course (strlen, …)
Overloading 1

- A function is defined by both its name **and** interface
  - double average (int a, int b)

- Can defined several (related) functions with the same name, as long as they have a unique interface
  - double average (int a, int b, int c)
  - double average (double a, double b)

- This is called **overloading** a function
Overloading 2

natural example

- double area (double length, double width)
- double area (double radius)
- double area (double xpenta[5], double ypenta[5])

in the context of overloading, interface = # of parameters and type of parameters
does not include return type

- so return type does not distinguish 2 functions
- int avg (int a, int b) collides with double avg(int a, int b)
Overloading 3

- can you think of a function that we have been using that is already overloaded?
- arithmetic operators
- division / is really a function
  - int / (int a, int b)
  - double / (double a, double b)
**Type conversion 1**

- you can avoid the use of overloading in many cases, because function arguments are converted to the correct type if necessary and possible
  - e.g., float a=3.1, b=5.7, c;  c = a/b;
  - promotes a and b to double so that we can use double division
  - don’t need to define float division, which behaves the same as double division
Type conversion 2

- suppose ‘double area (double length, double width)’ is defined
- don’t need to define ‘double area (float length, float width)’ or ‘double area (int length, float width)’ or ... (unless you want them to behave differently)
- OK to promote int to float, float to double
- average(1,2) returns 1.5
consider ‘int a; double b; b = foo(a);’
what happens?
- compiler looks for ‘double foo(int)’
- if it doesn’t find it, it doesn’t give up
  - looks for ‘double foo (float), perhaps
  - if it doesn’t find it, looks for ‘double foo (double)’
- suppose it finds this: promotes a to double and uses this function
- tries to find an interpretation that works
Day 20

- return of lab midterm
- discussion of lab midterm
- solution of lab midterm
- discussion of upcoming lecture midterm
- discussion of future lectures
- discussion of primes and cryptography
Day 21: Sieve of Erastothenes

- problem: find first \( p \) primes
- \( x \) is prime if \( x \) is an integer > 1 and the only factors of \( x \) are 1 and \( x \)
- store primes in an array
- consider integers in order 2, 3, 4, …
- as you consider an integer \( i \), mark its multiples 2\( i \), 3\( i \), …
- as you consider \( i \), if it is not yet marked, it is prime
Sieve 2

- as you consider i, if it IS marked, it is not prime and you don’t need to mark its multiples (why?)
- use an array to record marked integers
  - how do you mark an integer?
  - store mark for i in index i
    - yes, this wastes 2 array elements (0 and 1)
    - clarity of code is more important than space requirements
General algorithm for sieve

- error if \( p > \text{MAXNPRIME} \)
- initialize table
- for each table entry \( i \)
  - if not marked
    - add to list of primes
    - increment \# of primes found
    - if finished, return
    - mark multiples of \( i \)
Sieve implementation

do this in lab on Tuesday
Tennis score

- input: # of points won
- output: your score (to the screen)
- a big nested if-else statement
- is there a more elegant way?
- yes: switch statement (if all tests are of the same value, in this case # of points)
Switch statement

a concise way to write a long if-else that depends on a single value (of a variable, formula, …)

switch (value) {
    case val1: <some statements>; break;
    case val2: <some statements>; break;
    ...
    default: <some statements>;
}

Switch (cont.)

- some breaks may be missing
  - fall through to next case
- e.g., counting vowels
Day 22

Lecture midterm!