Homework 6

1. void letter\_grade(int points){

if(points > 89){

\*gradep = ‘A’;

}else if(points > 79){

\*gradep = ‘B’;

}else if(points > 69){

\*gradep = ‘C’;

}else if(points > 59){

\*gradep = ‘D’;

}else{

\*gradep = ‘F’

}

if(points % 10 == 9)

\*just\_missedp = 1;

}

1. When a function is called, memory space is assigned for each formal parameter called within the function. The value of each input parameter is stored in the memory allocated to its corresponding formal parameter. The value of the output parameter is stored in the memory cell in which the output parameter points to, through the address of the output parameter.
2. All functions shown can call grumpy.

/\* nonsense \*/

void silly(int \*x);

int main(void) {

int x, y;

x = 10; y = 11;

// x is now being mutated in silly

// but y in main is not

silly(&x);

// stop here

silly(&y);

/\* values here \*/

. . .

}

void silly(int \*x) {

int y;

// \*x references the variable x in main by address aka a pointer

y = \*x + 2;

\*x = 2 \* \*x;

}

* 1. A program written with function is commonly smaller in lines of code because if a piece of code is used repeatedly it is more effective to write it once and call it n times instead of writing it n times.
  2. Function subprogram errors are just as avoidable as any errors, the programmer must simply take a structured, top-down approach to avoid errors.