Function Arguments and Relational Operators

1. The following program is an answer to the last worksheet question from last class. How is yours different? How many versions of `celebrate` would have to be written to have one for everyone in the class? Redesign this function by using the point of variance (the part of the function that differs across versions) to determine the arguments the function should take to eliminate the redundancy of multiple versions. Edit the code given directly. Don't forget to update the function documentation to match your change!

```
#include <stdio.h>

void celebrate();

int main( void ) {
   celebrate(29);
   return 0;
}

/*
 * Purpose: prints happy birthday message followed by the given age
 * Parameters: int age - a valid age in years
 *
 */
void celebrate(int age) {
   ~~int age = 29;~~
   printf( "Happy Birthday!  You are %d\n", age);
}
```

2. Write and test a function that will take a floating point number and print double the value of that number. Try your best to write this without looking at Question 1.

```
#include <stdio.h>

void print_double(double n);

int main( void ) {
   print_double(5);

   return 0;
}

/*
 * Purpose: prints  the value of n multiplied by 2
 * Parameters: double n
 *
 */
void print_double (double n) {
   n *= 2;
   printf( "%f\n", n);
}
```

3. What is the output of the following program?

```
#include <stdio.h>

void fn1(int x);
void fn2();
void fn3(int x);
void fn4();

int main( void ) {
   fn2();
   fn4();
   return 0;
}

void fn1(int x) {
   printf( "fn1: %d\n", x);
   x++;
   printf( "fn1: %d\n", x);
}

void fn2( ) {
   int y = 5;
   printf( "fn2: %d\n", y);
   fn1(y);
   printf( "fn2: %d\n", y);
}

void fn3(int x) {
   printf( "fn3: %d\n", x);
   x *= 2;
   printf( "fn3: %d\n", x);
}

void fn4( ) {
   int x = 6;
   printf( "fn4: %d\n", x);
   fn3(x);
   printf( "fn4: %d\n", x);
}
```

```
fn2: 5
fn1: 5
fn1: 6
fn2: 5
fn4: 6
fn3: 6
fn3: 12
fn4: 6
```

4. For each of the code segment (i and ii):

If `data` is initialized to -2 before each code segment, what is the output when the code executes?

i) `data` has the value 4
ii) `data` has the value 2
iii) `data` has the value -2

If `data` is initialized to 10 before each code segment, what is the output when the code executes?

i) `data` has the value 10
ii) `data` has the value 20
iii) `data` has the value 10

i)

```
if( data < 0 ) {
   data *= -1;
}

if( data < 5 ) {
   data *= 2;
}
printf("data has the value %d\n", data);
```

ii)

```
if( data < 0 ) {
   data *= -1;
}
else {
   data *= 2;
}
printf("data has the value %d\n", data);
```

iii)

```
if( data < 0 ) {
   int data = 100;
   data *= -1;
}
else {
   int data = 200;
   data *= 2;
}
printf("data has the value %d\n", data);
```

5. You were asked to write a function that determines the cost of riding the bus based on the value of a variable `age` of type `int`. If `age` is less than 18, the cost is $1.50. If `age` is 65 or more, the cost is $2.00. For all other values of `age`, the cost is $2.50. A friend of yours submitted the following code:

```
void print_fare(int age);

int main( void ) {

    print_fare(17);
    print_fare(18);
    print_fare(65);
    print_fare(66);

    return 0;
}

/*
 * Purpose: determines the bus fare based on age and prints it
 * Parameters: int age - age in years, >=0
 */
void print_fare(int age) {
   double fare;

   if( age < 18 )
       fare = 1.50;
   if( age >= 65 )
       fare = 2.00;
   if( age >= 18 && age < 65 )
      fare = 2.50;

   printf("The fare is: $%4.2f\n", fare);

}
```

The instructor knocked off marks for "inappropriate use of branching constructs and redundant Boolean expressions and readability of magic numbers ". Rewrite the function in light of the comments.

```
#define CHILD_FARE 1.5
#define ADULT_FARE 2.5
#define SENIOR_FARE 2
#define ADULT  18
#define SENIOR 65

void print_fare(int age);

int main( void ) {

    print_fare(17);
    print_fare(18);
    print_fare(65);
    print_fare(66);

    return 0;
}

/*
 * Purpose: determines the bus fare based on age and prints it
 * Parameters: int age - age in years, >=0
 */
void print_fare(int age) {
   if( age < ADULT   )
       fare = CHILD_FARE;
   else if( age >= SENIOR )
       fare = SENIOR_FARE;
   else
      fare = ADULT_FARE;

   printf("The fare is: $%4.2\n", fare);

}
```

6. Design a function that will take an integer and print the value of that integer and whether it is odd or not. HINT: what does the % operator do?

```
#include <stdio.h>

void is_odd(int n);

int main(void) {

   is_odd(1); // should print 1 is odd
   is_odd(12); // should print 12 is even

   return 0;
}

/*
 * Purpose: prints whether n is odd or not
 * Parameters: int n
 */
void is_odd(int n) {
   if( n%2 == 0 )
      printf("%d is even\n", n);
   else
      printf("%d is odd\n", n);

}
```

7. Design a function that takes a number of adults, number of children, and number of seniors and prints the total cost for everyone to ride the bus. Assume that the number of people is not negative.

```
#include <stdio.h>

#define CHILD_FARE 1.5
#define ADULT_FARE 2.5
#define SENIOR_FARE 2

void total_cost(int adult, int child, int senior);

int main(void) {

   total_cost(3, 2, 1); // should print $12.50
   total_cost(1, 3, 2); // should print $11.00

   return 0;

}
/*
 * Purpose: prints whether n is odd or not
 * Parameters: int adult - number of adults, >=0
 *             int child - number of children, >=0
 *             int senior - number of seniors, >=0
 */
void total_cost(int adult, int child, int senior) {
   double cost = adult * ADULT_FARE + child * CHILD_FARE + senior * SENIOR_FARE;


   printf("Total fare is: $%.2f\n", cost);

}
```