

## 1) (10 pts) DSN (Recursive Functions)

Consider the problem of transforming a positive integer  $X$  into a positive integer  $Y$  when the only two operations you are allowed are adding 1 to the current number or multiplying the current number by 2. Write a recursive function that returns the minimum number of steps necessary to transform  $X$  to  $Y$ . If  $X > Y$ , return 1000000000, to indicate that no solution exists. (For example, if  $X = 13$  and  $Y = 28$ , the correct response would be 2 - first you would add 1 to 13 to obtain 14, then multiply 14 by 2 to obtain 28.) Feel free to call the provided function. *Note: don't worry about the run time of your function - assume that the inputs are such that the run time is relatively small, even when written using straight-forward recursion. There is a clever, efficient solution without recursion but please write the slower recursive solution since the goal of this question is to test recursive thinking.*

```
#define NO_SOLUTION 1000000000
```

```
int min(int x, int y) {  
    if (x < y) return x;  
    return y;  
}
```

```
// Returns the minimum number of steps to transform x into y, or  
// 1000000000 to indicate no solution.
```

```
int minSteps(int x, int y) {  
  
    if (x > y) return NO_SOLUTION;           // 2 pts  
    if (x == y) return 0;                    // 2 pts    int mult = 1 + minSteps(2*x, y);         // 2 pts  
    int add = 1 + minSteps(x+1, y);          // 2 pts    return min(add, mult);                   // 2 pts  
}
```