This is a open book, open notes, closed neighbor, closed teammate, closed laptop/cell phone/iPad/etc. exam. You may also use your calculator for this exam. Partial credit is available, so wherever possible, Show Your Work, particularly when tracing the execution of programs. When writing code, it is NOT necessary to comment the code on this exam, however, you should use meaningful variable and function names to make your code readable.

This exam consists of 7 numbered pages, including this cover sheet. You are advised to quickly look over the entire exam first to plan your time. You will have as much time as you need for this exam (within reason), but should be able to complete it in about an hour.
1. **Trace Me** (25 Points). Trace the following program and show the exact output produced. Show all of your work - just the answer is not sufficient.

```c
#include <stdio.h>
#define BASE 2
int bit(int, int);
int pow(int, int);

main()
{
    int x, y;
    int b, n;
    x = 5;
    y = 4;
    n = 2;

    printf("%d = ", x);
    while (n >= 0)
    {
        b = bit(x, n);
        x = x - b * y;
        printf("%d ", b);
        y = y / BASE;
        n = n - 1;
    }
    printf("\n");
}

int bit(int a, int b)
{
    int x;
    x = pow(BASE, b);
    if(a >= x)
        return 1;
    else return 0;
}

int pow(int x, int y)
{
    int a = 1;
    while (y > 0)
    {
        a = x * a;
        y = y - 1;
    }
    return a;
}
2. Fix Me (20 Points). The other day some telemarketer called you and offered you a “... deal you cannot refuse”. He said if you invest by the end of the week, he can double your money in 3 years!

Your friend decides to check out his claim and writes a program which will determine the least whole number interest rate which will double an initial investment in 3 years. Assume annual compounding. But the program, given below, seems to have some problems. Your job is to find and fix as many errors as you can. Identify whether each error would be found at compile, link or run time.

```c
#include stdio.h

int compound(float start, float rate, int time)

main
{
    float initial, final;
    int interest

    flag = scanf("%f", &initial);
    while(initial != EOF)
    {
        final = initial;
        intr = 1;
        while(final < 2 * initial)
            final = compound(initial , interest/100 , three);
        interest = interest + 1;

        printf("Initial investment of %4.2f at %d %% yields %4.2f" 
                " in 3 years\n initial, interest ");
        sanf("%f", initial);
    }
}

float compound_it(float amt, float rate, int intervals);
{ float amt;

    while( intervals > 0 );
    {
        amt = amt + amt * rate;
        intervals = intervals - 1;

        return amt;
    }
```
3. **Orbit prediction**

Last Spring the UH CubeSat team got their first CubeSat into orbit and they asked us to help them track the satellite so the ground station crew knows when to tune their receiver to listen for the satellite. They can give us the exact time (hrs, mins, secs) that the satellite crossed our longitude line (called a zenith time), and would like a program that can tell them the next three zenith times (if they know that, they can figure out when to start listening).

They told us that the satellite is in a 90 minute orbit, but since the earth (and our longitude line) also rotates another 22.5 degrees in the 90 minutes, the satellite must travel another 5 minutes 38 seconds to reach the zenith again. So the effective orbit time is 1:35:38.

We will use the five step process to design this program.

(a) **Understand the problem:** The program is given the exact time of the last zenith pass and should print the next three zenith times (in 24 hour time). For example, if the last zenith time was 10:17:00, the program should print:

```
Enter zenith time (hr min sec, EOF to quit): 10 17 00
Next zenith at 11:52:38
Next zenith at 13:28:16
Next zenith at 15:03:54
Enter zenith time (hr min sec, EOF to quit):
```

(b) (10 Points). **Do a hand example:**

Show how you would calculate the above results for this example.
(c) (15 Points). Write the algorithm:

Write the algorithm for main for this program.
(d) (20 Points). **Translate the algorithm to C**

Before we write the code for main, one observation we have is that we need to add two values modulo 60 in two places (for seconds and minutes) so we consider factoring that out into a function. However, the function has to return the result of the addition, mod 60, and whether or not there is a carry. For example, 35 + 42 is 17 with a carry of 1, while 18 + 22 is 40 with a carry of 0. However, we all know a function can return only one thing as the value of the function! But your TA has a clever idea to return this extra information in the value of the function. The result of adding two values between 0 and 59 will only have a carry of either 0 or 1, so we can have the function return the positive value if the carry is 0, and the negative of the value if the carry is 1. The prototype for the function is:

```c
int add_mod_60(int a, int b);
/* Given: two values between 0 and 59
   adds a and b modulo 60
   Returns: the result if no carry, negative of the result if there is a carry
*/
```

Write the code for main() using this function as needed.
(e) (10 Points).
Write the code for the function add_mod_60().