1. (Target-Heart-Rate Calculator) While exercising, you can use a heart-rate monitor to see that your heart rate stays within a safe range suggested by your trainers and doctors. According to the American Heart Association (AHA), the formula for calculating your maximum heart rate in beats per minute is 220 minus your age in years. Your target heart rate is a range that's 50–85% of your maximum heart rate. [Note: These formulas are estimates provided by the AHA. Maximum and target heart rates may vary based on the health, fitness and gender of the individual. Always consult a physician or qualified health care professional before beginning or modifying an exercise program.] Create a program that reads the user's birthday and the current day (each consisting of the month, day and year). Your program should calculate and display the person's age (in years), the person's maximum heart rate and the person's target- heart-rate range.

```c
#include <stdio.h>

int main() {
    //initializations
    int birthMonth, birthDay, birthYear;
    int currentMonth, currentDay, currentYear;
    int currentAge, maxHeartRate;
    double targetHeartRateLow, targetHeartRateHigh;

    // Get the birth date
    printf("Enter your birth date MM DD YYYY: ");
    scanf("%d %d %d", &birthMonth, &birthDay, &birthYear);

    // Get current date
    printf("Enter the current date MM DD YYYY: ");
    scanf("%d %d %d", &currentMonth, &currentDay, &currentYear);

    // Calculate age
    currentAge = currentYear - birthYear;
    if (currentMonth < birthMonth || (currentMonth == birthMonth && currentDay < 
birthDay)) {
        currentAge--; // birthday hasn't occurred yet this year
    }

    // Calculate the maximum heart rate
    maxHeartRate = 220 - currentAge;

    // Calculate target heart rate range
    targetHeartRateLow = maxHeartRate * 0.50;
    targetHeartRateHigh = maxHeartRate * 0.85;
```

```
    // Display results
    printf("\nYour current age is: %d years old\n", currentAge);
    printf("Your reccomended maximum heart rate is: %d bpm\n", maxHeartRate);
    printf("Your target heart rate range is : %.1f to %.1f bpm\n",
targetHeartRateLow, targetHeartRateHigh);

    scanf("%d",birthMonth); //Stalls for the .exe
    return 0;
}
```

Source code.

Example 1.

Birthday: 06/16/2003
Todays date: 05/13/2025

```
PS D:\Git Repos\CIS241>  & 'c:\Users\luket\.vscode\extensions\ms-vscode.cpptool
osoft-MIEngine-Pid-4mvuimnf.xea' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--in
Enter your birth date MM DD YYYY: 06 16 2003
Enter the current date MM DD YYYY: 05 13 2025

Your current age is: 21 years old
Your reccomended maximum heart rate is: 199 bpm
Your target heart rate range is : 99.5 to 169.2 bpm
```

Example 2.

Birthday 01/01/2000

Todays date 05/13/2025

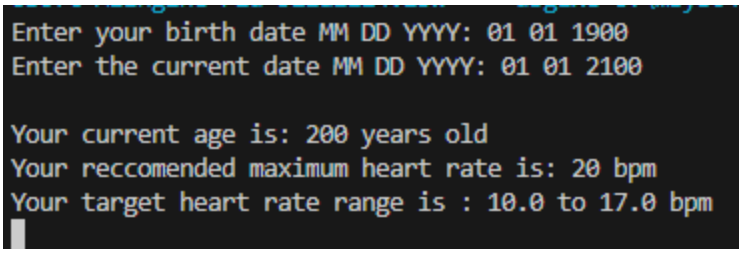
```
Enter your birth date MM DD YYYY: 01 01 2000
Enter the current date MM DD YYYY: 05 13 2025

Your current age is: 25 years old
Your reccomended maximum heart rate is: 195 bpm
Your target heart rate range is : 97.5 to 165.8 bpm
```

Example 3

Birthday 01/01/1900

Todays date 01/01/2100



2. (Improving the Recursive Fibonacci Implementation) In Section 5.15, the recursive algorithm we used to calculate Fibonacci numbers was intuitively appealing. However, recall that the algorithm resulted in the exponential explosion of recursive function calls. Research the recursive Fibonacci implementation online. Study the various approaches, including the iterative version in Exercise 5.35 and versions that use only so- called "tail recursion." Discuss the relative merits of each.

   In order to look at the three different ways to explore the Fibonacci implementation, lets look at each one individually. We can also look at the big O notation of each of the methods in order to find out the complexity of them. Methods with a smaller big O notation will be better due to the fact that they will be more efficient.

   Recursive Implementation – We know that this has an exponential big O notation meaning the bigger number that we put into a function that uses the recursive implementation will have greater effects on the end product due to it getting more and more complicated the more iterations it has to self reference. It has to hold more and more instances of the function on the stack before they can resolve. It also does the same calculations many times, repeating work is never good.

   Iterative Implementation – This is not as flashy as recursive Implementation but it is much simpler and can handle larger numbers without a problem. By using variables to store they values every time it goes through it does not hold any additional instances of the function on the stack.

   Tail Recursion Implementation – In Tail recursion, it seems like the function will only use the final recursive element at the very end of the call, this will result in the best of both worlds with the speed and flashiness of the recursive implementation, and the space benefits of the iterative implementation. It is both very fast while also being very memory efficient.