```c
/*
Assignment #4: See how single and double precision values are
               represented for an infinite series.

Author: Shawn Hinnebusch

Date: 10/09/2020

Double Precision: gcc -DDOUBLEPREC -O3 -o hw4.exe  main.c -lm
Single Precision: gcc -O3 -o hw4.exe  main.c -lm

Create PDF: a2ps main.c --pro=color --columns=2 -E -o hw04.ps | ps2pdf hw04.ps


*/
#include "timer.h"
#include <float.h>
#include <limits.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/resource.h>

#ifdef DOUBLEPREC
typedef double   REAL;
typedef long int INT;
#define LOOP_MAX 1.5e12 // Max iterations to take an hour
#define MACHINE_PRECISION DBL_EPSILON

#else
typedef float REAL;
typedef int   INT;
#define LOOP_MAX INT_MAX
#define MACHINE_PRECISION FLT_EPSILON
#endif

void exponentialFunctin(INT a);
void problem1Part1( );
void problem1Part2( );

int main( )
{
    printf("Largest positive integer that can be represented on a 64-bit machine: %llu\n", ULLONG_MAX);
    printf("Epsilon in single precision: %e\n", FLT_EPSILON);
    printf("Epsilon in double precision: %e\n", DBL_EPSILON);

    // ###################### Problem 1 ####################################
    problem1Part1( );
    problem1Part2( );

    // ###################### Problem 2 ####################################
    // Define vector of values for problem 2
    INT a[ 10 ] = {1, 5, 10, 15, 20, -1, -5, -10, -15, -20};

    FILE *output;
#ifdef DOUBLEPREC
    output = fopen("Problem2Double.dat", "w");
    if (!output) exit(1);
    fprintf(output, "x\tResidual\tTaylor\t\t\texp(x)\t\t\tIterations\n");
#else
    output = fopen("Problem2Single.dat", "w");
    if (!output) exit(1);
    fprintf(output, "x\tResidual\tTaylor\t\texp(x)\t\tIterations\n");
#endif
    fclose(output);

    for (int i = 0; i < 10; i++) {
        exponentialFunctin(a[ i ]);
    }
```

```c
    return 0;
}

// Functions

// ##################### Problem 1 Part 1 ###################################
void problem1Part1( )
{
    REAL   sum = 0;
    REAL   residual;
    double start, finish, elapsedTime;
    // Output the results to the file

    FILE *outputPart1;
#ifdef DOUBLEPREC
    outputPart1 = fopen("Problem1Part1Double.dat", "w");
    fprintf(outputPart1, "Double Precision: Infinite Series summation of 1/n\n");
#else
    outputPart1 = fopen("Problem1Part1Single.dat", "w");
    fprintf(outputPart1, "Single Precision: Infinite Series summation of 1/n\n");
#endif

    if (!outputPart1) exit(1);

    fprintf(outputPart1, "Iteration\tResidual\tsum\n");

    GET_TIME(start);
    INT n = 1;
    for (n = 1; n < LOOP_MAX; n++) {
        sum      = sum + 1.0 / ( REAL ) n;
        residual = (1.0 / ( REAL ) n) / sum;

        if (residual < MACHINE_PRECISION) { break; }

#ifdef DOUBLEPREC
        if (n % 10000000 == 0) { fprintf(outputPart1, "%12ld\t%e\t%10.15lf\n", n, res
idual, sum); }
#else
        if (n % 10000 == 0) { fprintf(outputPart1, "%d\t\t%e\t%9.6f\n", n, residual,
 sum); }
#endif
    }

#ifdef DOUBLEPREC
    fprintf(outputPart1, "%12ld\t%e\t%10.15lf\n", n, residual, sum);
#else
    fprintf(outputPart1, "%d\t\t%e\t%9.6f\n", n, residual, sum);
#endif

    // Finish time and display time
    GET_TIME(finish);
    elapsedTime = finish - start;
    fprintf(outputPart1, "Total time 1/n = %f seconds\n", elapsedTime);

    fclose(outputPart1);
}

// ##################### Problem 1 Part 2 ###################################
void problem1Part2( )
{
    REAL   sum = 0;
    REAL   residual;
    double start, finish, elapsedTime;

    GET_TIME(start);
    // Output the results to the file
    FILE *output;
#ifdef DOUBLEPREC
```

```
    output = fopen("Problem1Part2Double.dat", "w");
    if (!output) exit(1);
    fprintf(output, "Double Precision: Infinite Series summation of 1/n^2\n");
#else
    output = fopen("Problem1Part2Single.dat", "w");
    if (!output) exit(1);
    fprintf(output, "Single Precision: Infinite Series summation of 1/n^2\n");
#endif
    fprintf(output, "Iteration\tResidual\tsum\n");
    INT n;
    for (n = 1; n < LOOP_MAX; n++) {
        sum      = sum + 1.0 / (n * n);
        residual = (1.0 / (n * n)) / sum;

        if (residual < MACHINE_PRECISION) { break; }
        // Output results to file
#ifdef DOUBLEPREC
        if (n % 10000 == 0) {
            fprintf(output, "%10ld\t%e\t%10.15lf\n", n, residual, sum);
#else
        if (n % 100 == 0) {
            fprintf(output, "%d\t\t%e\t%9.6f\n", n, residual, sum);
#endif
        }
    } // end of Loop

// Print final values after convergence
#ifdef DOUBLEPREC
    fprintf(output, "%10ld\t%e\t%10.15lf\n", n, residual, sum);
#else
    fprintf(output, "%d\t\t%e\t%9.6f\n", n, residual, sum);
#endif
    GET_TIME(finish);
    elapsedTime = finish - start;
    fprintf(output, "Total time 1/n^2= %.6f seconds\n", elapsedTime);

    fclose(output);
}

void exponentialFunctin(INT factNum)
{
    // Define Variables
    REAL residual = 1;
    INT  n;
    REAL stoppingCrit;
    REAL expNum    = 1;
    REAL expNumNeg = 1;
    for (n = 1; n <= LOOP_MAX; n++) {
        // Split into positive or negative to use a better numerical
        // solution to calculate e^-x with better accuracy
        if (factNum > 0) {
            residual     = residual * factNum / n;
            expNum       = expNum + residual;
            stoppingCrit = fabs(residual) / fabs(expNum);
        } else {
            residual     = residual * (-1) * factNum / n;
            expNumNeg    = expNumNeg + residual;
            expNum       = 1 / expNumNeg;
            stoppingCrit = fabs(residual) / fabs(expNum);
        }

        if (stoppingCrit < MACHINE_PRECISION) { break; }
    }

    // Output results to file
    FILE *output;
#ifdef DOUBLEPREC
    output = fopen("Problem2Double.dat", "a");
    if (!output) exit(1);
```

```
    fprintf(output, "%ld\t%e\t%.15e\t%.15e\t%ld\n", factNum, residual, expNum, exp(fact
Num), n);
#else
    output = fopen("Problem2Single.dat", "a");
    if (!output) exit(1);
    fprintf(output, "%d\t%e\t%.6e\t%.6e\t%d\n", factNum, residual, expNum, exp(factNu
m), n);
#endif
    fclose(output);
}
```