Name: Antonio Santos

Lab Section: B03

Course: ENCM 369

Lab Assignment: Lab 1

Exercise G

// exG.c

// ENCM 369 Winter 2019 Lab 1 Exercise G

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX\_ABS\_F (5.0e-12)

#define POLY\_DEGREE 4

*double* polyval(const *double* \*a, *int* n, *double* x);

// Return a[0] + a[1] \* x + ... + a[n] \* pow(x, n).

*int* main(*void*)

{

*double* f[ ] = {1.5, 0.7, -3.1, -1.2, 1.0};

*double* dfdx[POLY\_DEGREE];

*double* guess;

*int* max\_updates;

*int* update\_count;

*int* scan\_count;

*int* i;

*int* close\_enough = 0;

*double* current\_x, current\_f, current\_dfdx;

printf("This program demonstrates use of Newton's Method to find\n"

"approximate roots of the polynomial\nf(x) = ");

printf("%.2f", f[0]);

i = 1;

first\_for\_start:

if(i > POLY\_DEGREE) goto first\_for\_end;

if(f[i] < 0) goto first\_else\_code;

printf(" + %.2f\*pow(x,%d)", f[i], i);

i++;

goto first\_for\_start;

first\_else\_code:

printf(" - %.2f\*pow(x,%d)", -f[i], i);

i++;

goto first\_for\_start;

first\_for\_end: ;

printf("\nPlease enter a guess at a root, and a maximum number of\n"

"updates to do, separated by a space.\n");

scan\_count = scanf("%lf%d", &guess, &max\_updates);

if(scan\_count == 2) goto correct\_count;

printf("Sorry, I couldn't understand the input.\n");

exit(1);

correct\_count: ;

if(max\_updates >= 0) goto valid\_input;

printf("Sorry, a negative limit on updates does not make sense.\n");

exit(1);

valid\_input: ;

printf("Running with initial guess %f.\n", guess);

i = POLY\_DEGREE - 1;

second\_for\_start:

if(i < 0) goto second\_for\_end;

dfdx[i] = (i + 1) \* f[i + 1];

i--;

goto second\_for\_start;

second\_for\_end: ;

current\_x = guess;

update\_count = 0;

while\_loop\_start:

if(!1) goto while\_loop\_end;

current\_f = polyval(f, POLY\_DEGREE, current\_x);

printf("%d update(s) done; x is %.15f; f(x) is %.15e\n",

update\_count, current\_x, current\_f);

if(fabs(current\_f) > MAX\_ABS\_F) goto end\_first\_if;

close\_enough = 1;

goto while\_loop\_end;

end\_first\_if: ;

if(update\_count != max\_updates) goto end\_second\_if;

goto while\_loop\_end;

end\_second\_if: ;

current\_dfdx = polyval(dfdx, POLY\_DEGREE - 1, current\_x);

current\_x -= current\_f / current\_dfdx;

update\_count++;

goto while\_loop\_start;

while\_loop\_end: ;

if(!close\_enough) goto second\_else\_code;

printf("Stopped with approximate solution of %.12f.\n",

current\_x);

goto end\_if;

second\_else\_code:

printf("%d updates performed, |f(x)| still >= %g.\n",

update\_count, MAX\_ABS\_F);

goto end\_if;

end\_if: ;

return 0;

}

*double* polyval(const *double* \*a, *int* n, *double* x)

{

*double* result = a[n];

*int* i;

i = n - 1;

for\_loop\_start:

if(i < 0) goto for\_loop\_end;

result = x \* result + a[i];

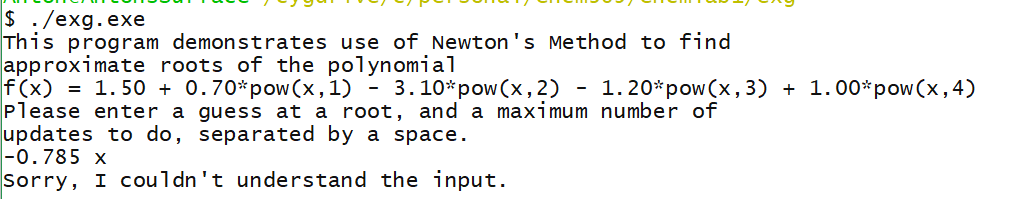
i--;

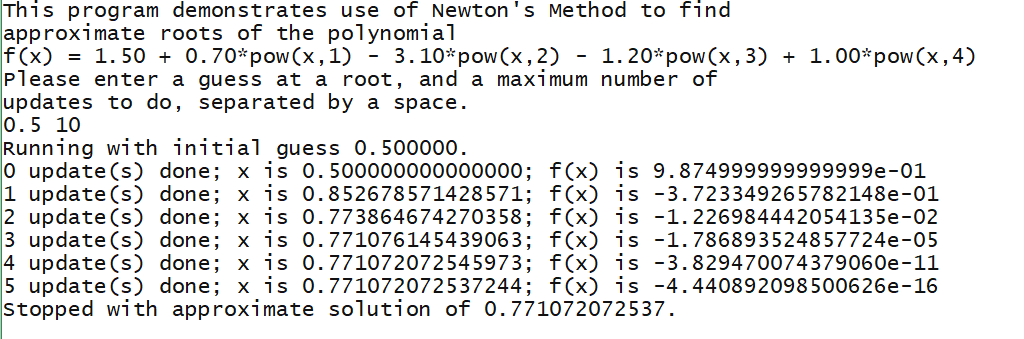
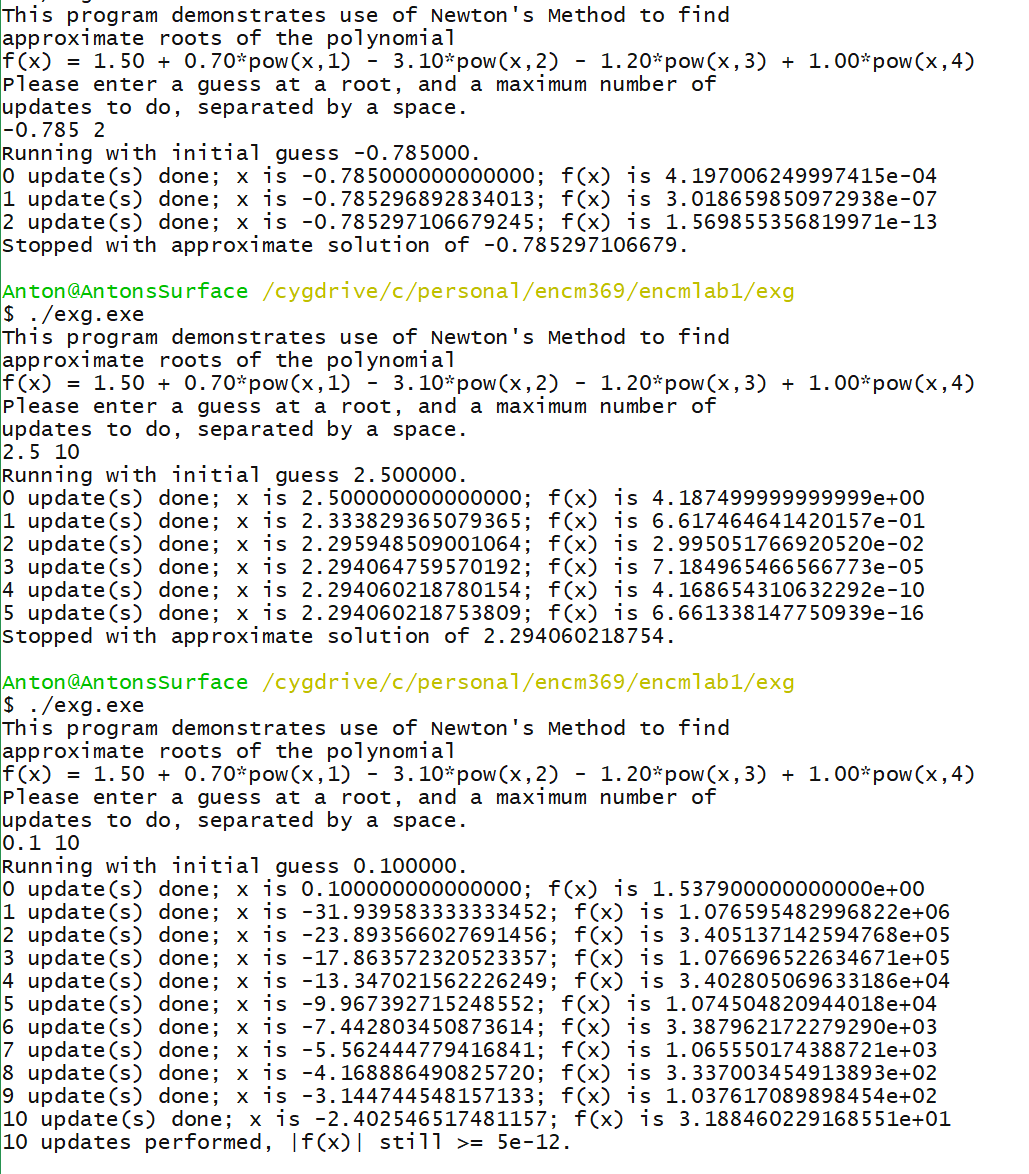
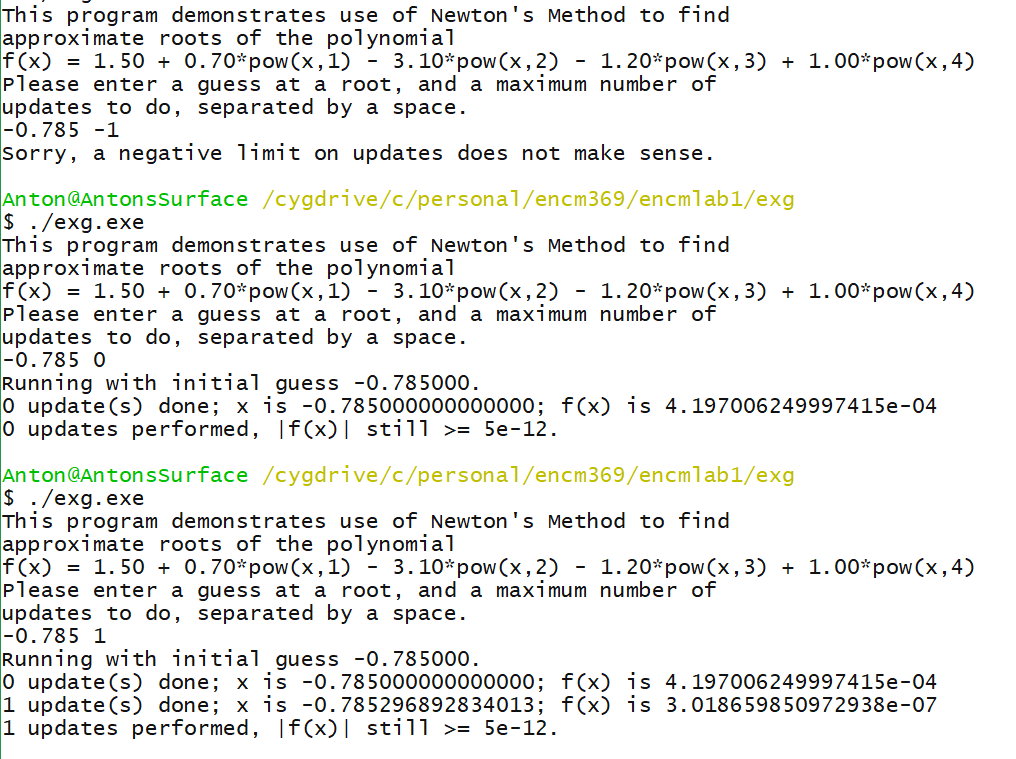
goto for\_loop\_start;

for\_loop\_end: ;

return result;

}







Exercise H

// exH.c

// ENCM 369 Winter 2019 Lab 1 Exercise H

#include <stdio.h>

*void* print\_array(const *char* \*str, const *int* \*a, *int* n);

// Prints the string given by str on stdout, then

// prints a[0], a[1], ..., a[n - 1] on stdout on a single line.

*void* sort\_array(*int* \*a, *int* n);

// Sorts a[0], a[1], ..., a[n - 1] from smallest to largest.

*int* main(*void*)

{

*int* test\_array[] = {440, 220, 330, 550, 330, 660, 110, 330, 440};

print\_array("before sorting ...", test\_array, 9);

sort\_array(test\_array, 9);

print\_array("after sorting ...", test\_array, 9);

return 0;

}

*void* print\_array(const *char* \*str, const *int* \*a, *int* n)

{

*int* i;

puts(str);

i = 0;

for\_loop\_start:

if(i >= n) goto for\_loop\_end;

printf(" %d", a[i]);

i++;

goto for\_loop\_start;

for\_loop\_end: ;

printf("\n");

}

*void* sort\_array(*int* \*a, *int* n)

{

// This is an implementation of an algorithm called selection sort.

*int* i, j, min, j\_of\_min;

i = 0;

outer\_for\_start:

if(i >= n-1) goto outer\_for\_end;

min = a[i];

j\_of\_min = i;

j = i + 1;

inner\_for\_start:

if(j >= n) goto inner\_for\_end;

if(a[j] >= min) goto end\_if;

min = a[j];

j\_of\_min = j;

end\_if: ;

j++;

goto inner\_for\_start;

inner\_for\_end: ;

a[j\_of\_min] = a[i];

a[i] = min;

i++;

goto outer\_for\_start;

outer\_for\_end: ;

}

