**Exercise G: (Parts I edited are bolded for easier viewing)**

// 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;**

**loop1\_Top:**

**if(i > POLY\_DEGREE) goto endLoop1;**

**if(f[i] < 0) goto next;**

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

**goto endif1;**

**next:**

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

**endif1:**

**i++;**

**goto loop1\_Top;**

**endLoop1:**

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 endif2;**

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

**exit(1);**

**endif2:**

**if (max\_updates >= 0) goto endif3;**

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

**exit(1);**

**endif3:**

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

**i = POLY\_DEGREE - 1;**

**loop2\_Top:**

**if(i < 0) goto endLoop2;**

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

**current\_x = guess;**

**update\_count = 0;**

**i--;**

**goto loop2\_Top;**

**endLoop2:**

**loop3\_Top:**

**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 endif4;**

**close\_enough = 1;**

**goto endLoop3;**

**endif4:**

**if(update\_count != max\_updates) goto endif5;**

**goto endLoop3;**

**endif5:**

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

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

**update\_count++;**

**goto loop3\_Top;**

**endLoop3:**

**if (!close\_enough) goto next2;**

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

**current\_x);**

**goto endif6;**

**next2:**

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

**update\_count, MAX\_ABS\_F);**

**endif6:**

return 0;

}

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

{

*double* result = a[n];

***int* i = n - 1;**

**loop4\_Top:**

**if(i < 0) goto endLoop4;**

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

**i--;**

**goto loop4\_Top;**

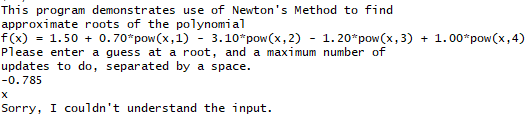
**endLoop4:**

return result;

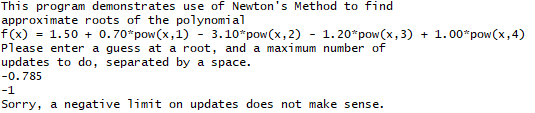
}

**Outputs:**

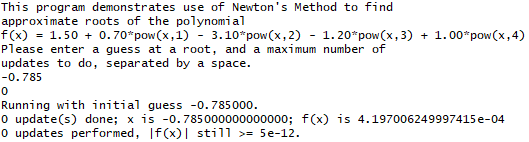
Run 1: -0.785 x



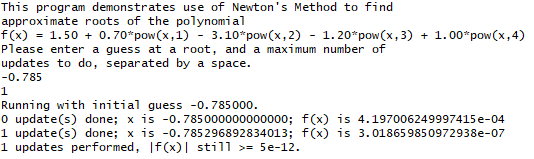
Run 2: -0.785 -1



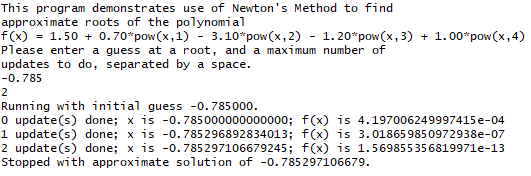
Run 3: -0.785 0



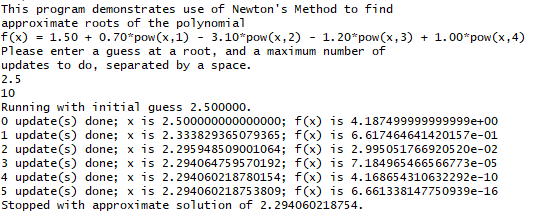
Run 4: -0.785 1



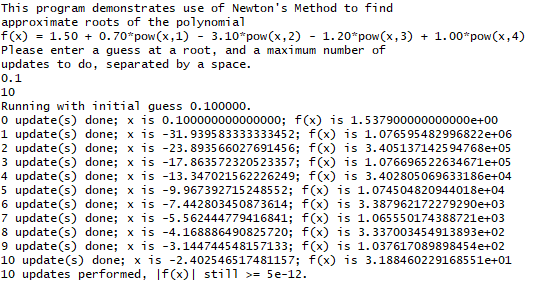
Run 5: -0.785 2



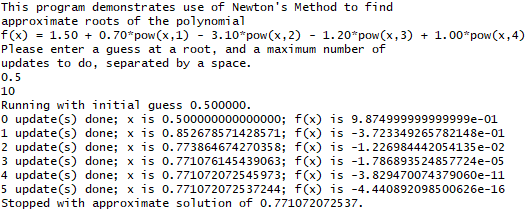
Run 6: 2.5 10



Run 7: 0.1 10



Run 8: 0.5 10



**Exercise H: (Parts I edited are bolded for easier viewing)**

// 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;**

**loop\_Top:**

**if(i >= n) goto endLoop;**

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

**i++;**

**goto loop\_Top;**

**endLoop:**

**printf("\n");**

}

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

{

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

**i = 0;**

**loop2\_Top:**

**if( i >= n-1) goto endLoop2;**

**min = a[i];**

**j\_of\_min = i;**

**j = i + 1;**

**loop3\_Top:**

**if(j >= n) goto endLoop3;**

**if(a[j] >= min) goto endif;**

**min = a[j];**

**j\_of\_min = j;**

**endif:**

**j++;**

**goto loop3\_Top;**

**endLoop3:**

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

**a[i] = min;**

**i++;**

**goto loop2\_Top;**

**endLoop2:**

**;**

}

**Output:**

