Course ​​​: Computer Organization – ENCM 369

Lab # ​​​: Lab 1

Student Name ​: Nimna Wijedasa

Lab Section ​​: B04

Ex B

DiagramDiagram, schematic

Description automatically generated

Table

Table

Description automatically generated

Ex G

Source code

// lab1exG.c

// ENCM 369 Winter 2023 Lab 1 Exercise G

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX\_ABS\_F (5.0e-9)

#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[] = {1.47, 0.73, -2.97, -1.15, 1.00};

double dfdx[POLY\_DEGREE];

double guess;

int max\_updates;

int update\_count;

int n\_scanned;

int i;

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;

loop\_top\_1:

if (!(i <= POLY\_DEGREE)) goto past\_end\_0;

if (f[i] >= 0) goto else\_code\_1;

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

goto end\_if\_1;

else\_code\_1:

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

end\_if\_1:

i++;

goto loop\_top\_1;

past\_end\_0:

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

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

n\_scanned = scanf("%lf%d", &guess, &max\_updates);

if (!(n\_scanned != 2)) goto end\_1;

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

exit(1);

end\_1:

;

if (!(max\_updates < 1)) goto end\_2;

printf("Sorry, I must be allowed do at least one update.\n");

exit(1);

end\_2:

;

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

for (i = POLY\_DEGREE - 1; i >= 0; i--)

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

current\_x = guess;

update\_count = 0;

int k = 1;

loop\_top\_in:

if (k == 0) goto past\_end\_1;

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 past\_end\_1;

if (update\_count == max\_updates)

goto past\_end\_1;

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

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

update\_count++;

goto loop\_top\_in;

past\_end\_1:

if (fabs(current\_f) >= MAX\_ABS\_F) goto else\_code\_2;

printf("Stopped with approximate solution of %.10f.\n",

current\_x);

goto end\_if\_5;

else\_code\_2:

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

update\_count, MAX\_ABS\_F);

end\_if\_5:

return 0;

}

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

{

double result = a[n];

int i;

i = n - 1;

loop\_top\_2:

if (!(i >= 0)) goto past\_end\_2;

result \*= x;

result += a[i];

i--;

goto loop\_top\_2;

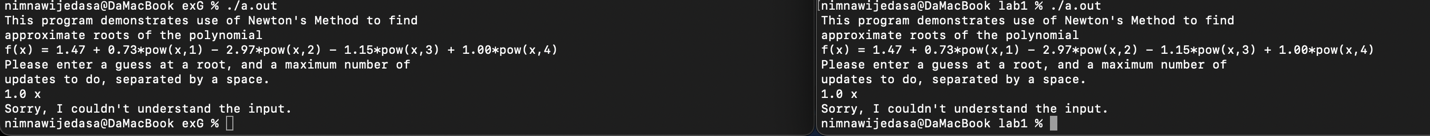
past\_end\_2:

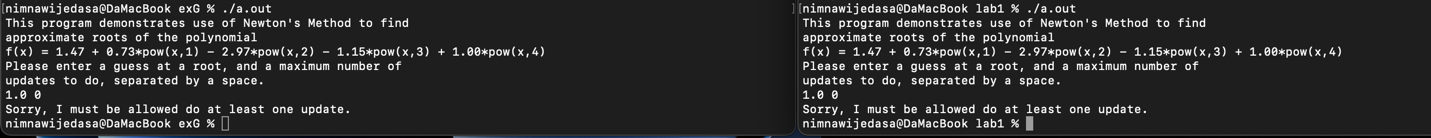
return result;

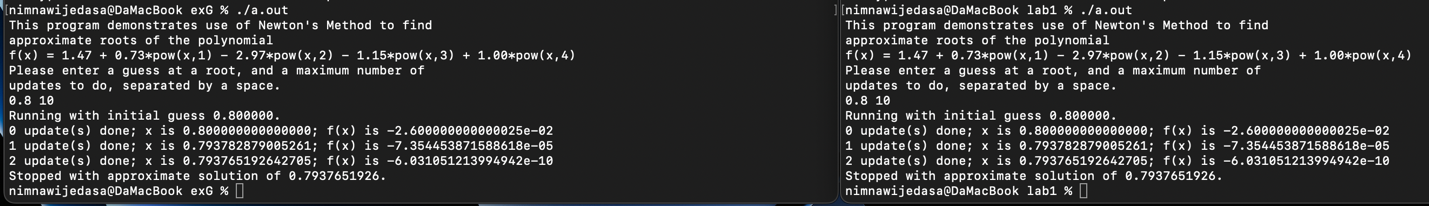
}

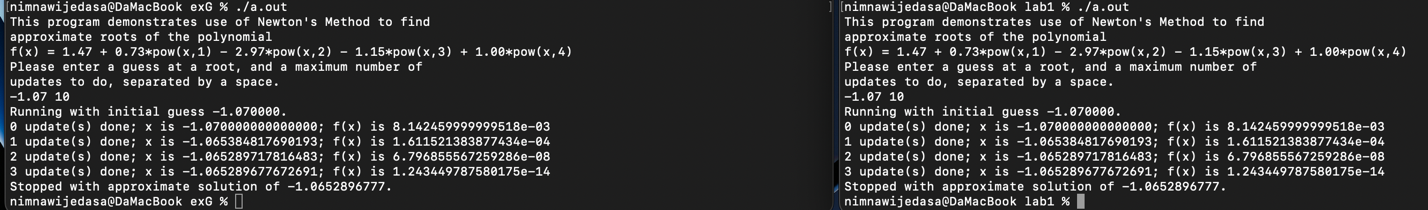
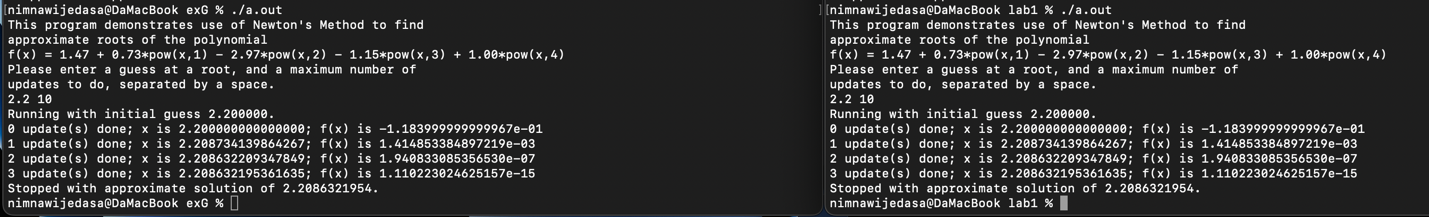
Test runs

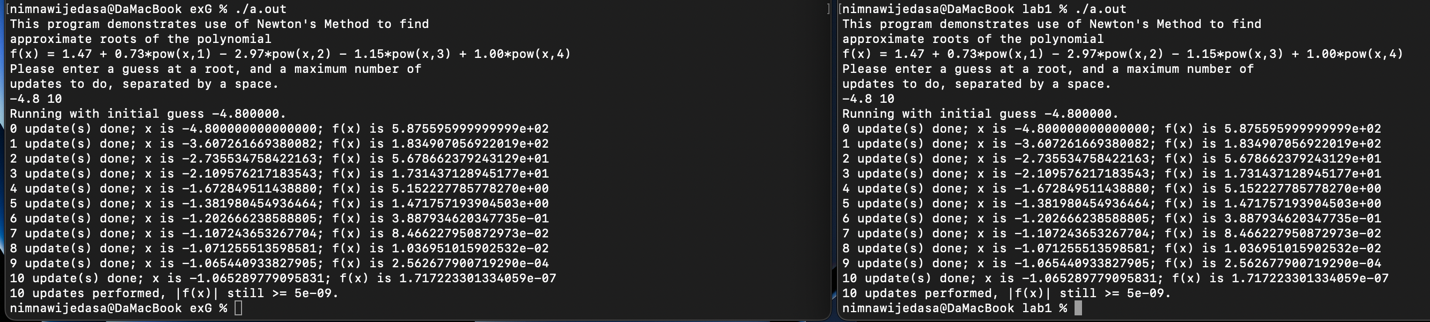
Left is goto C code that was converted from the C code which is on the right







A screenshot of a computer

Description automatically generated with medium confidence

Ex H

// lab1exH.c

// ENCM 369 Winter 2023 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 \*x, int n);

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

int main(void)

{

int test\_array[] = { 4000, 5000, 7000, 1000, 3000, 4000, 2000, 6000 };

print\_array("before sorting ...", test\_array, 8);

sort\_array(test\_array, 8);

print\_array("after sorting ...", test\_array, 8);

return 0;

}

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

{

int i = 0;

puts(str);

start:

if (!(i < n)) goto end;

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

printf("\n");

i++;

goto start;

end:

;

};

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

{

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

int outer = 1, inner, vti;

outer\_loop:

if (!(outer < n)) goto quit\_outer\_loop;

vti = x[outer];

inner = outer;

inner\_loop:

if (!(inner > 0)) goto quit\_inner\_loop;

if (!(vti < x[inner - 1])) goto quit\_inner\_loop;

x[inner] = x[inner - 1];

inner--;

goto inner\_loop;

quit\_inner\_loop:

x[inner] = vti;

outer++;

goto outer\_loop;

quit\_outer\_loop:;

};

Text

Description automatically generated