**Course**: ENCM 369

**Lab Section:** B03

**Lab 1**

**Student Name**: Mitchell Sawatzky

**Date Submitted**: Jan 22, 2016

Exercise 5.2

Point Three

|  |  |  |
| --- | --- | --- |
| Static | Stack | Heap |
| |  |  | | --- | --- | | aa[0] | 210 | | aa[1] | 321 | | aa[2] | 432 | | aa[3] | 543 |  |  |  | | --- | --- | | bb[0] | 4004 | | bb[1] | 3003 | | bb[2] | 2002 | | bb[3] | 1001 | |  |  |  |  |  | | --- | --- | | cc | 635 | | |  | | --- | | AR reverse | | |  |  | | --- | --- | | guard | • |  |  |  | | --- | --- | | k | 0 | | | |  |  | | --- | --- | | dest | • |  |  |  | | --- | --- | | src | • |  |  |  | | --- | --- | | n | 4 | |  |  | | --- | | AR main | | |  |  | | --- | --- | | dd[0] | 210 | | dd[1] | 321 | | dd[2] | 432 | | dd[3] | 543 | | dd[4] | ?? | | dd[5] | ?? |  |  |  | | --- | --- | | ee[0] | 1001 | | ee[1] | 2002 | | ee[2] | 3003 | | ee[3] | 4004 | | | no args | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| instant in time | dest | src | guard |
| point two, first time | 0x10003C | 0x7FFE60 | 0x10004C |
| point two, second time | 0x100040 | 0x7FFE60 | 0x10004C |
| point two, third time | 0x100044 | 0x7FFE60 | 0x10004C |
| point two, fourth time | 0x100048 | 0x7FFE60 | 0x10004C |

Exercise 12.1

exG.c

// exG.c

// ENCM 369 Winter 2016 Lab 1 Exercise G

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX\_ERROR (0.5e-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[] = { 1.45, 0.78, -3.03, -1.15, 1.00 };

double dfdx[POLY\_DEGREE];

double guess;

int update\_max;

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;

start\_for\_b:

if (i > POLY\_DEGREE)

goto end\_for\_b;

if (f[i] >= 0)

goto if\_a;

goto el\_a;

if\_a:

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

goto fi\_a;

el\_a:

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

goto fi\_a;

fi\_a:

i++;

goto start\_for\_b;

end\_for\_b:

;

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, &update\_max);

if (n\_scanned != 2)

goto if\_b;

goto el\_b;

if\_b:

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

exit(0);

el\_b:

;

if (update\_max < 1)

goto if\_c;

goto el\_c;

if\_c:

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

exit(0);

el\_c:

;

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

i = 0;

start\_for\_c:

if (i >= POLY\_DEGREE)

goto end\_for\_c;

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

i++;

goto start\_for\_c;

end\_for\_c:

;

current\_x = guess;

update\_count = 0;

start\_while\_a:

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 (update\_count == update\_max)

goto end\_while\_a;

if (fabs(current\_f) < MAX\_ERROR)

goto end\_while\_a;

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

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

update\_count++;

goto start\_while\_a;

end\_while\_a:

;

if (fabs(current\_f) <= MAX\_ERROR)

goto if\_d;

goto el\_d;

if\_d:

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

current\_x);

goto fi\_d;

el\_d:

printf("%d updates performed, solution still not good enough.\n",

update\_count);

goto fi\_d;

fi\_d:

;

return 0;

}

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

{

double result = 0.0;

int i = n;

start\_for\_a:

if (i < 1)

goto end\_for\_a;

result += a[i];

result \*= x;

i--;

goto start\_for\_a;

end\_for\_a:

;

result += a[0];

return result;

}

Terminal Output:

|  |  |
| --- | --- |
| Before Modification | After Modification |
| Mitchell@ttys003 21:23 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 x  Sorry, I couldn't understand the input.  Mitchell@ttys003 21:23 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 0  Sorry, I must be allowed do at least one update.  Mitchell@ttys003 21:23 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 3  Running with initial guess 1.000000.  0 update(s) done; x is 1.000000000000000; f(x) is -9.499999999999995e-01  1 update(s) done; x is 0.799154334038055; f(x) is -4.082592009647401e-02  2 update(s) done; x is 0.789490702493091; f(x) is -1.844257849812347e-04  3 update(s) done; x is 0.789446648301911; f(x) is -3.908826595733217e-09  3 updates performed, solution still not good enough.  Mitchell@ttys003 21:23 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 4  Running with initial guess 1.000000.  0 update(s) done; x is 1.000000000000000; f(x) is -9.499999999999995e-01  1 update(s) done; x is 0.799154334038055; f(x) is -4.082592009647401e-02  2 update(s) done; x is 0.789490702493091; f(x) is -1.844257849812347e-04  3 update(s) done; x is 0.789446648301911; f(x) is -3.908826595733217e-09  4 update(s) done; x is 0.789446647368162; f(x) is -4.440892098500626e-16  Stopped with approximate solution of 0.7894466474.  Mitchell@ttys003 21:23 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.6 10  Running with initial guess 1.600000.  0 update(s) done; x is 1.600000000000000; f(x) is -3.215599999999999e+00  1 update(s) done; x is -0.757478005865107; f(x) is -5.033601557507139e-02  2 update(s) done; x is -0.727014138172744; f(x) is 2.690899366527333e-03  3 update(s) done; x is -0.728488476178189; f(x) is 5.772146689952962e-06  4 update(s) done; x is -0.728491652366615; f(x) is 2.690980771546947e-11  Stopped with approximate solution of -0.7284916524.  Mitchell@ttys003 21:23 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.8 10  Running with initial guess 1.800000.  0 update(s) done; x is 1.800000000000000; f(x) is -3.172399999999998e+00  1 update(s) done; x is 3.368941641938672; f(x) is 5.453299577903110e+01  2 update(s) done; x is 2.789754341062644; f(x) is 1.564652553251245e+01  3 update(s) done; x is 2.433108167336762; f(x) is 3.892116274400187e+00  4 update(s) done; x is 2.265542885662034; f(x) is 6.369755201980263e-01  5 update(s) done; x is 2.225371164494491; f(x) is 3.168430569887160e-02  6 update(s) done; x is 2.223154162819966; f(x) is 9.333252154331007e-05  7 update(s) done; x is 2.223147593508815; f(x) is 8.180014443581740e-10  8 update(s) done; x is 2.223147593451238; f(x) is -3.108624468950438e-15  Stopped with approximate solution of 2.2231475935.  Mitchell@ttys003 21:23 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  -4.9 10  Running with initial guess -4.900000.  0 update(s) done; x is -4.900000000000000; f(x) is 6.366541500000002e+02  1 update(s) done; x is -3.682586792591736; f(x) is 1.988315861602772e+02  2 update(s) done; x is -2.792782092071352; f(x) is 6.152307486560385e+01  3 update(s) done; x is -2.154146355902563; f(x) is 1.873771511369693e+01  4 update(s) done; x is -1.709692969524729; f(x) is 5.550965265339927e+00  5 update(s) done; x is -1.416516273962120; f(x) is 1.560088387483382e+00  6 update(s) done; x is -1.241764686358543; f(x) is 3.889185355548186e-01  7 update(s) done; x is -1.158552879036898; f(x) is 6.926890137589847e-02  8 update(s) done; x is -1.135842956725481; f(x) is 4.584744310158761e-03  9 update(s) done; x is -1.134112442806661; f(x) is 2.581306211357770e-05  10 update(s) done; x is -1.134102588756783; f(x) is 8.350702351833661e-10  10 updates performed, solution still not good enough.  Mitchell@ttys003 21:24 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  -4.8 10  Running with initial guess -4.800000.  0 update(s) done; x is -4.800000000000000; f(x) is 5.859171999999999e+02  1 update(s) done; x is -3.609082335341513; f(x) is 1.828922472361626e+02  2 update(s) done; x is -2.739520499139934; f(x) is 5.654175045866470e+01  3 update(s) done; x is -2.116476165436881; f(x) is 1.719481254798258e+01  4 update(s) done; x is -1.684131848769382; f(x) is 5.080194869870039e+00  5 update(s) done; x is -1.400420498862172; f(x) is 1.419965705529570e+00  6 update(s) done; x is -1.233078423204510; f(x) is 3.491090297747030e-01  7 update(s) done; x is -1.155372672226102; f(x) is 5.965980569434337e-02  8 update(s) done; x is -1.135439304126264; f(x) is 3.516731278879748e-03  9 update(s) done; x is -1.134108413976470; f(x) is 1.525952628567140e-05  10 update(s) done; x is -1.134102588549394; f(x) is 2.918416619479558e-10  Stopped with approximate solution of -1.1341025885. | Mitchell@ttys001 22:18 {130} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 x  Sorry, I couldn't understand the input.  Mitchell@ttys001 22:18 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 0  Sorry, I must be allowed do at least one update.  Mitchell@ttys001 22:18 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 3  Running with initial guess 1.000000.  0 update(s) done; x is 1.000000000000000; f(x) is -9.499999999999995e-01  1 update(s) done; x is 0.799154334038055; f(x) is -4.082592009647401e-02  2 update(s) done; x is 0.789490702493091; f(x) is -1.844257849812347e-04  3 update(s) done; x is 0.789446648301911; f(x) is -3.908826595733217e-09  3 updates performed, solution still not good enough.  Mitchell@ttys001 22:18 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.0 4  Running with initial guess 1.000000.  0 update(s) done; x is 1.000000000000000; f(x) is -9.499999999999995e-01  1 update(s) done; x is 0.799154334038055; f(x) is -4.082592009647401e-02  2 update(s) done; x is 0.789490702493091; f(x) is -1.844257849812347e-04  3 update(s) done; x is 0.789446648301911; f(x) is -3.908826595733217e-09  4 update(s) done; x is 0.789446647368162; f(x) is -4.440892098500626e-16  Stopped with approximate solution of 0.7894466474.  Mitchell@ttys001 22:19 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.6 10  Running with initial guess 1.600000.  0 update(s) done; x is 1.600000000000000; f(x) is -3.215599999999999e+00  1 update(s) done; x is -0.757478005865107; f(x) is -5.033601557507139e-02  2 update(s) done; x is -0.727014138172744; f(x) is 2.690899366527333e-03  3 update(s) done; x is -0.728488476178189; f(x) is 5.772146689952962e-06  4 update(s) done; x is -0.728491652366615; f(x) is 2.690980771546947e-11  Stopped with approximate solution of -0.7284916524.  Mitchell@ttys001 22:19 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  1.8 10  Running with initial guess 1.800000.  0 update(s) done; x is 1.800000000000000; f(x) is -3.172399999999998e+00  1 update(s) done; x is 3.368941641938672; f(x) is 5.453299577903110e+01  2 update(s) done; x is 2.789754341062644; f(x) is 1.564652553251245e+01  3 update(s) done; x is 2.433108167336762; f(x) is 3.892116274400187e+00  4 update(s) done; x is 2.265542885662034; f(x) is 6.369755201980263e-01  5 update(s) done; x is 2.225371164494491; f(x) is 3.168430569887160e-02  6 update(s) done; x is 2.223154162819966; f(x) is 9.333252154331007e-05  7 update(s) done; x is 2.223147593508815; f(x) is 8.180014443581740e-10  8 update(s) done; x is 2.223147593451238; f(x) is -3.108624468950438e-15  Stopped with approximate solution of 2.2231475935.  Mitchell@ttys001 22:19 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  -4.9 10  Running with initial guess -4.900000.  0 update(s) done; x is -4.900000000000000; f(x) is 6.366541500000002e+02  1 update(s) done; x is -3.682586792591736; f(x) is 1.988315861602772e+02  2 update(s) done; x is -2.792782092071352; f(x) is 6.152307486560385e+01  3 update(s) done; x is -2.154146355902563; f(x) is 1.873771511369693e+01  4 update(s) done; x is -1.709692969524729; f(x) is 5.550965265339927e+00  5 update(s) done; x is -1.416516273962120; f(x) is 1.560088387483382e+00  6 update(s) done; x is -1.241764686358543; f(x) is 3.889185355548186e-01  7 update(s) done; x is -1.158552879036898; f(x) is 6.926890137589847e-02  8 update(s) done; x is -1.135842956725481; f(x) is 4.584744310158761e-03  9 update(s) done; x is -1.134112442806661; f(x) is 2.581306211357770e-05  10 update(s) done; x is -1.134102588756783; f(x) is 8.350702351833661e-10  10 updates performed, solution still not good enough.  Mitchell@ttys001 22:19 {0} [exG]$ ./test.out  This program demonstrates use of Newton's Method to find  approximate roots of the polynomial  f(x) = 1.45 + 0.78\*pow(x,1) - 3.03\*pow(x,2) - 1.15\*pow(x,3) + 1.00\*pow(x,4)  Please enter a guess at a root, and a maximum number of  updates to do, separated by a space.  -4.8 10  Running with initial guess -4.800000.  0 update(s) done; x is -4.800000000000000; f(x) is 5.859171999999999e+02  1 update(s) done; x is -3.609082335341513; f(x) is 1.828922472361626e+02  2 update(s) done; x is -2.739520499139934; f(x) is 5.654175045866470e+01  3 update(s) done; x is -2.116476165436881; f(x) is 1.719481254798258e+01  4 update(s) done; x is -1.684131848769382; f(x) is 5.080194869870039e+00  5 update(s) done; x is -1.400420498862172; f(x) is 1.419965705529570e+00  6 update(s) done; x is -1.233078423204510; f(x) is 3.491090297747030e-01  7 update(s) done; x is -1.155372672226102; f(x) is 5.965980569434337e-02  8 update(s) done; x is -1.135439304126264; f(x) is 3.516731278879748e-03  9 update(s) done; x is -1.134108413976470; f(x) is 1.525952628567140e-05  10 update(s) done; x is -1.134102588549394; f(x) is 2.918416619479558e-10  Stopped with approximate solution of -1.1341025885. |

Exercise 13.1

exH.c

// exH.c

// ENCM 369 Winter 2016 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\_for\_a:

if (i >= n)

goto end\_for\_a;

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

i++;

goto start\_for\_a;

end\_for\_a:

;

printf("\n");

}

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

{

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

int outer = 1;

int inner;

int v;

start\_for\_b:

if (outer >= n)

goto end\_for\_b;

v = x[outer];

inner = outer;

start\_while\_a:

if (inner <= 0)

goto end\_while\_a;

if (v >= x[inner-1])

goto end\_while\_a;

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

inner--;

goto start\_while\_a;

end\_while\_a:

;

x[inner] = v;

outer++;

goto start\_for\_b;

end\_for\_b:

;

}