Name: Aarsh Shah

Student Name: Aarsh Shah

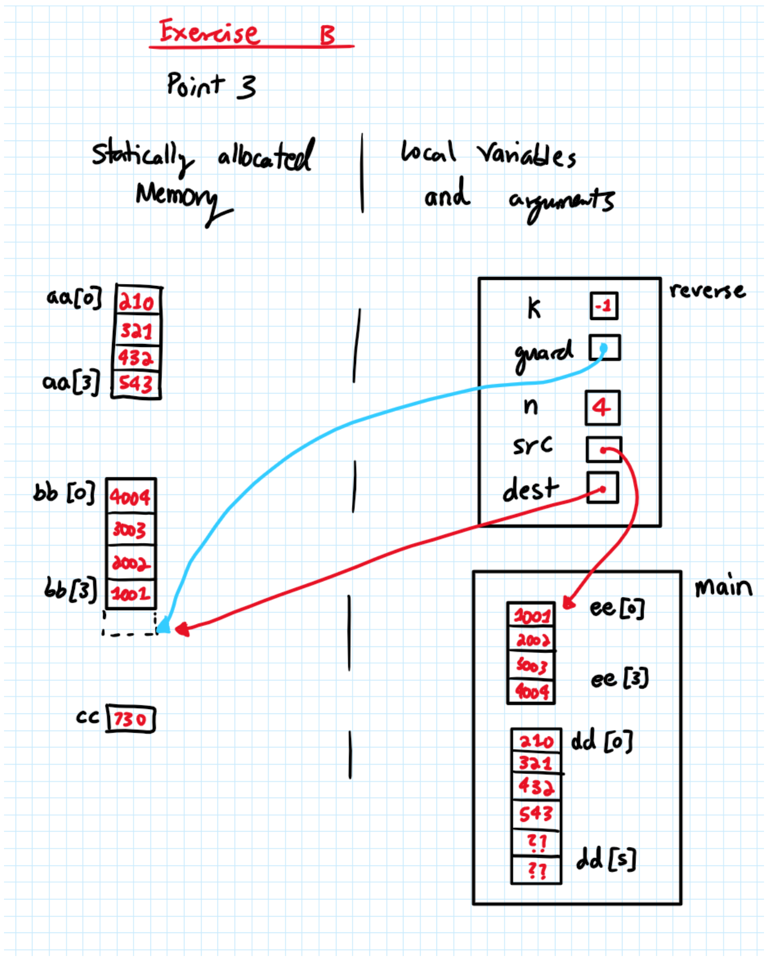
Lab Section: B02

Course: Computer Organization – ENCM 369

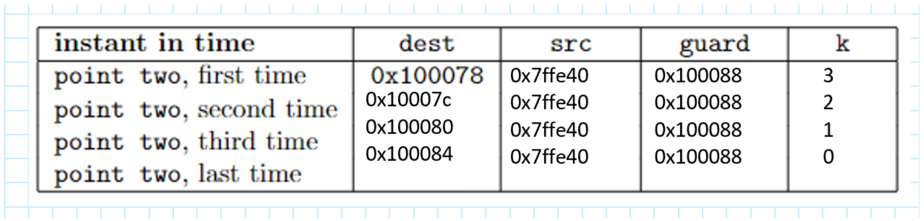
Lab #: 1

Exercise A – Built in Arrays in C

Part 1:



Part 2:



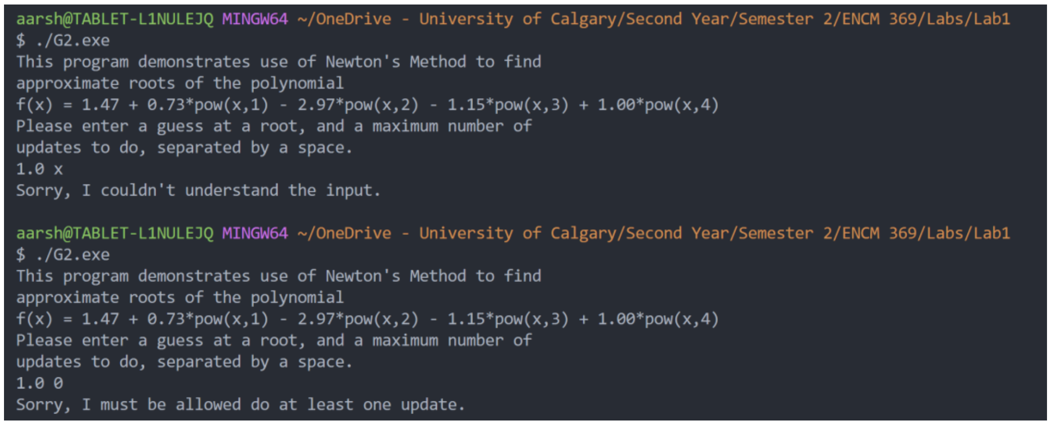
instant in time	dest	src	guard	k
point two, first time	0x100078	0x7ffe40	0x100088	3
point two, second time	0x10007c	0x7ffe40	0x100088	2
point two, third time	0x100080	0x7ffe40	0x100088	1
point two, last time	0x100084	0x7ffe40	0x100088	0

Exercise G: A more complicated program in Goto-C

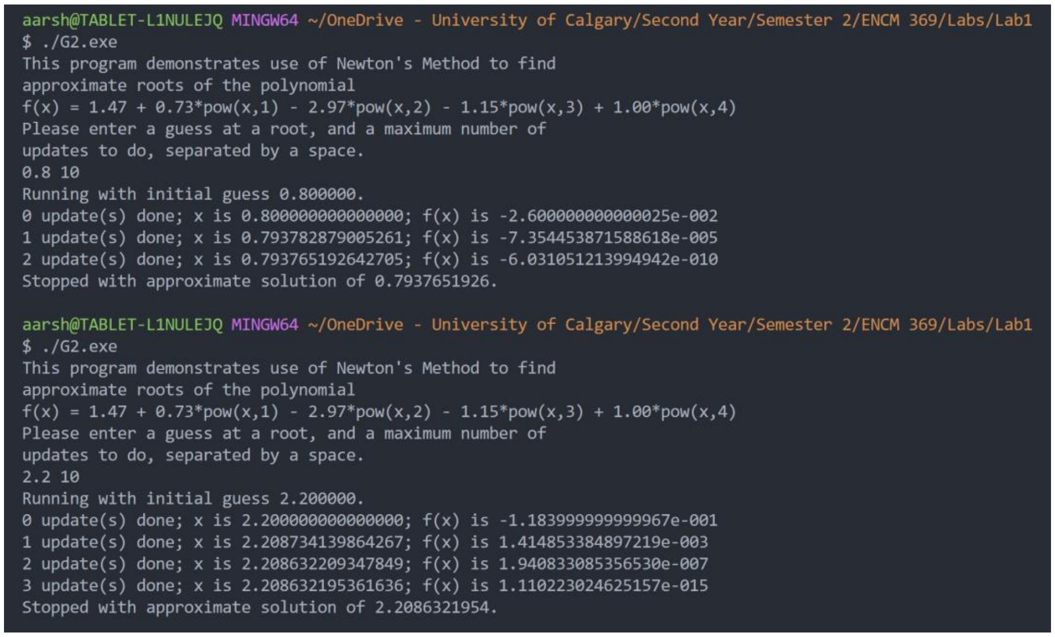
Source Code:

```c
// 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;
for_top1:
    if (i > POLY_DEGREE)
        goto for_end1;
    if (f[i] < 0)
        goto else_block1;
    printf(" + %.2f*pow(x,%d)", f[i], i);
    i++;
    goto for_top1;
else_block1:
    printf(" - %.2f*pow(x,%d)", -f[i], i);
    i++;
    goto for_top1;

for_end1:

    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 next10;
    printf("Sorry, I couldn't understand the input.\n");
    exit(1);

next10:
    if (max_updates >= 1)
        goto next11;
    printf("Sorry, I must be allowed do at least one update.\n");
    exit(1);

next11:
    printf("Running with initial guess %f.\n", guess);
    i = POLY_DEGREE - 1;

for_start2:
    if (i < 0)
        goto for_end2;
    dfdx[i] = (i + 1) * f[i + 1]; // Calculus!
    i--;
    goto for_start2;
for_end2:

    current_x = guess;
    update_count = 0;

while_start1:

    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 while_end1;
    if (update_count == max_updates)
        goto while_end1;
    current_dfdx = polyval(dfdx, POLY_DEGREE - 1, current_x);
    current_x -= current_f / current_dfdx;
    update_count++;
    goto while_start1;

while_end1:

    if (fabs(current_f) < MAX_ABS_F)
        goto else_if1;
    printf("%d updates performed, |f(x)| still >= %g.\n",
           update_count, MAX_ABS_F);
    goto end_of_func;
else_if1:
    printf("Stopped with approximate solution of %.10f.\n",
           current_x);

end_of_func:

    return 0;
}

double polyval(const double *a, int n, double x)
{
    double result = a[n];
    int i;

    i = n - 1;
for_start3:
    if (i < 0)
        goto for_end3;
    result *= x;
    result += a[i];
    i--;
    goto for_start3;

for_end3:
    return result;
}
```

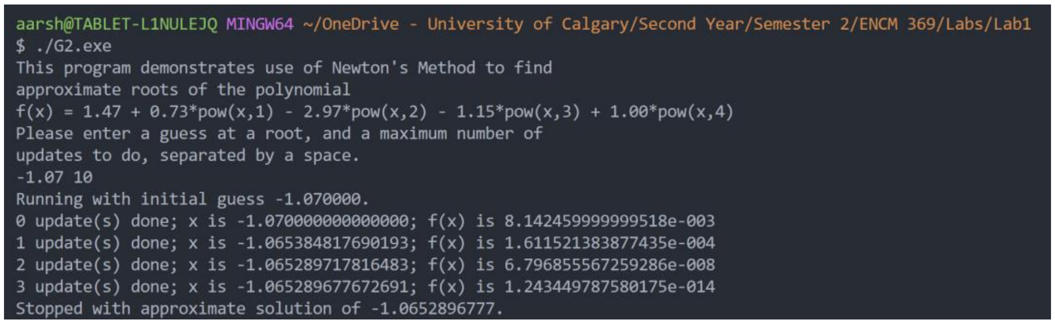
Screenshots:



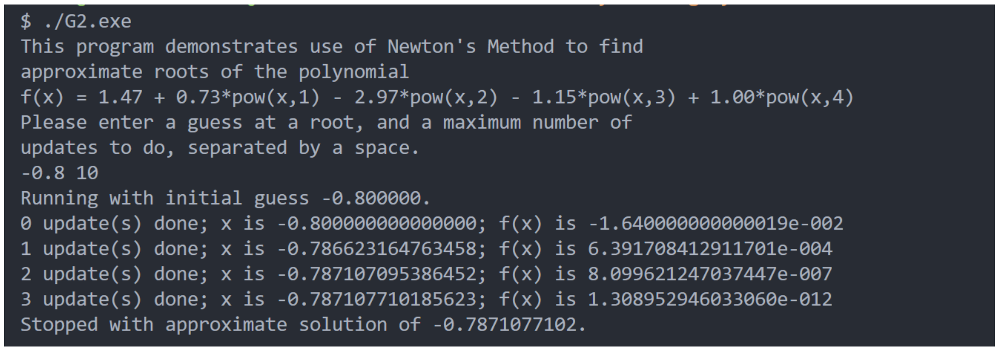
```
aarsh@TABLET-L1NULEJQ MINGW64 ~/OneDrive - University of Calgary/Second Year/Semester 2/ENCM 369/Labs/Lab1
$ ./G2.exe
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter 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.

aarsh@TABLET-L1NULEJQ MINGW64 ~/OneDrive - University of Calgary/Second Year/Semester 2/ENCM 369/Labs/Lab1
$ ./G2.exe
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*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.
```



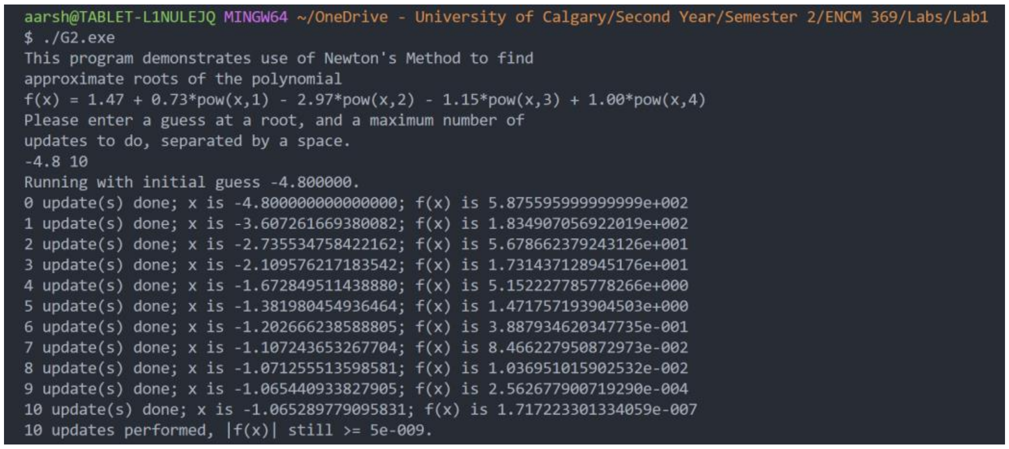
```
aarsh@TABLET-L1NULEJQ MINGW64 ~/OneDrive - University of Calgary/Second Year/Semester 2/ENCM 369/Labs/Lab1
$ ./G2.exe
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
0.8 10
Running with initial guess 0.800000.
0 update(s) done; x is 0.800000000000000; f(x) is -2.600000000000025e-002
1 update(s) done; x is 0.793782879005261; f(x) is -7.354453871588618e-005
2 update(s) done; x is 0.793765192642705; f(x) is -6.031051213994942e-010
Stopped with approximate solution of 0.7937651926.

aarsh@TABLET-L1NULEJQ MINGW64 ~/OneDrive - University of Calgary/Second Year/Semester 2/ENCM 369/Labs/Lab1
$ ./G2.exe
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*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.
2.2 10
Running with initial guess 2.200000.
0 update(s) done; x is 2.200000000000000; f(x) is -1.183999999999967e-001
1 update(s) done; x is 2.208734139864267; f(x) is 1.414853384897219e-003
2 update(s) done; x is 2.208632209347849; f(x) is 1.940833085356530e-007
3 update(s) done; x is 2.208632195361636; f(x) is 1.110223024625157e-015
Stopped with approximate solution of 2.2086321954.
```



```
aarsh@TABLET-L1NULEJQ MINGW64 ~/OneDrive - University of Calgary/Second Year/Semester 2/ENCM 369/Labs/Lab1
$ ./G2.exe
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*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.
-1.07 10
Running with initial guess -1.070000.
0 update(s) done; x is -1.070000000000000; f(x) is 8.142459999999518e-003
1 update(s) done; x is -1.065384817690193; f(x) is 1.611521383877435e-004
2 update(s) done; x is -1.065289717816483; f(x) is 6.796855567259286e-008
3 update(s) done; x is -1.065289677672691; f(x) is 1.243449787580175e-014
Stopped with approximate solution of -1.0652896777.
```



```
$ ./G2.exe
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*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.
-0.8 10
Running with initial guess -0.800000.
0 update(s) done; x is -0.800000000000000; f(x) is -1.640000000000019e-002
1 update(s) done; x is -0.786623164763458; f(x) is 6.391708412911701e-004
2 update(s) done; x is -0.787107095386452; f(x) is 8.099621247037447e-007
3 update(s) done; x is -0.787107710185623; f(x) is 1.308952946033060e-012
Stopped with approximate solution of -0.7871077102.
```



```
aarsh@TABLET-L1NULEJQ MINGW64 ~/OneDrive - University of Calgary/Second Year/Semester 2/ENCM 369/Labs/Lab1
$ ./G2.exe
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*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 5.875595999999999e+002
1 update(s) done; x is -3.607261669380082; f(x) is 1.834907056922019e+002
2 update(s) done; x is -2.735534758422162; f(x) is 5.678662379243126e+001
3 update(s) done; x is -2.109576217183542; f(x) is 1.731437128945176e+001
4 update(s) done; x is -1.672849511438880; f(x) is 5.152227785778266e+000
5 update(s) done; x is -1.381980454936464; f(x) is 1.471757193904503e+000
6 update(s) done; x is -1.202666238588805; f(x) is 3.887934620347735e-001
7 update(s) done; x is -1.107243653267704; f(x) is 8.466227950872973e-002
8 update(s) done; x is -1.071255513598581; f(x) is 1.036951015902532e-002
9 update(s) done; x is -1.065440933827905; f(x) is 2.562677900719290e-004
10 update(s) done; x is -1.065289779095831; f(x) is 1.717223301334059e-007
10 updates performed, |f(x)| still >= 5e-009.
```

Exercise H:

Source Code:

```c
// 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;
    puts(str);

    i = 0;
for_start1:
    if (i >= n)
        goto for_end1;
    printf("    %d", a[i]);
    i++;
    goto for_start1;

for_end1:

    printf("\n");
}

void sort_array(int *x, int n)
```

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

    int outer, inner, vti;

    outer = 1;
for_start2:
    if (outer >= n)
        goto for_end2;
    vti = x[outer];
    inner = outer;

while_start1:
    if (inner <= 0)
        goto while_end;
    if (vti >= x[inner - 1])
        goto while_end;
    x[inner] = x[inner - 1];
    inner--;
    goto while_start1;

while_end:
    x[inner] = vti;
    outer++;
    goto for_start2;

for_end2:;
}
```