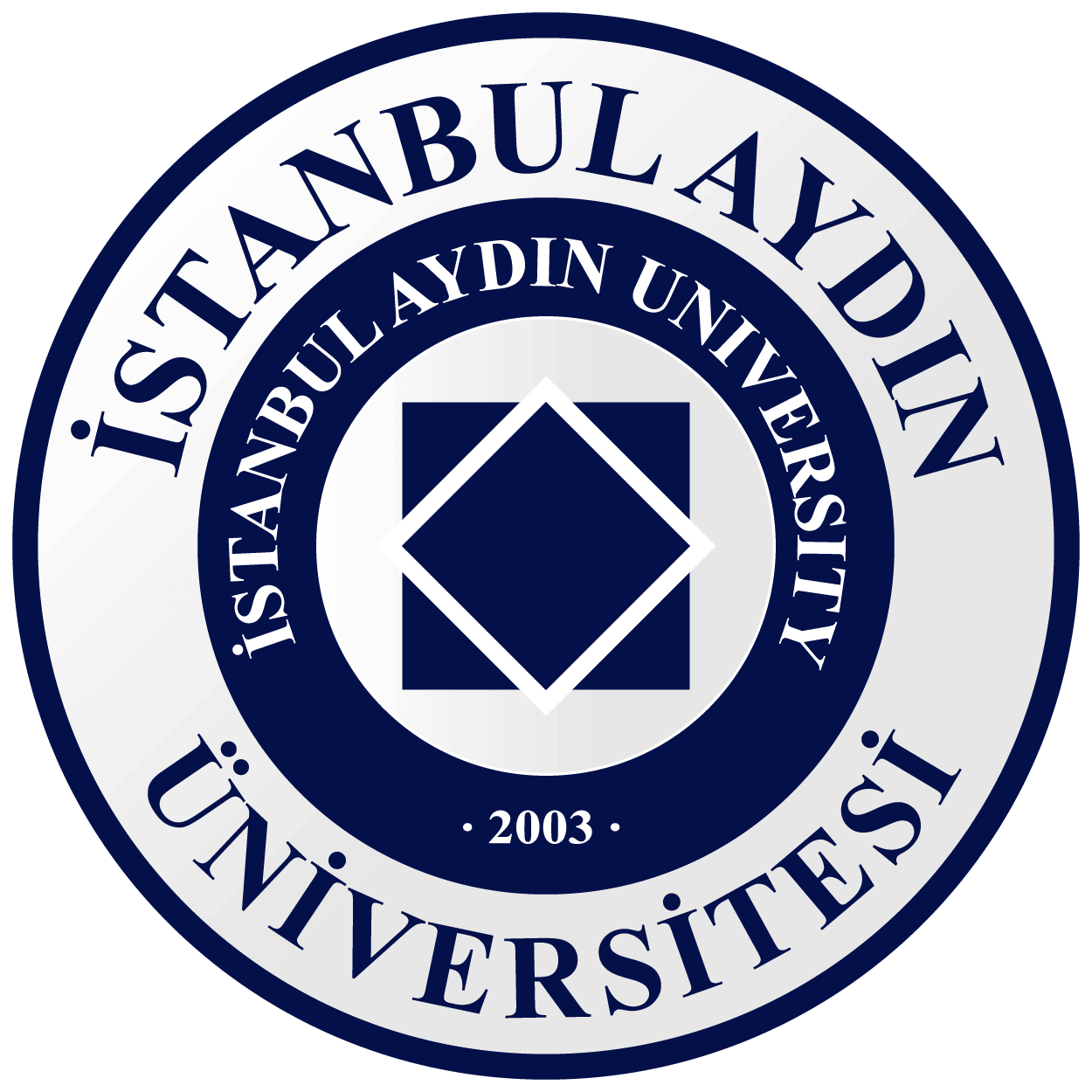
This document contains the contents of the project work for computer programming course, pertaining 40 points from the finals.

Final Exam Project:



Name: Mohammad Rauf

Student Number: B2105.010002

Department: Computer Engineering

Subject Code: COM117

Subject Name: COMPUTER PROGRAMMING-I

The proceeding explanations will be a detailed analysis of the project work. The build of the project is a makeshift calculator that does basic functions but also covers some extensive topics as well. Albeit not the best one that could have been made but also fitting the requirement as much as possible, covering all the topics taken up until these points:

The code is comprised in the following format:

1. Libraries inclusion.
2. Naming functions.
3. Main function where all the respective functions are called.
4. Ending main.
5. Started identifying and creating the named functions.

This program was created for the sole purpose of making an all-in-one convenient working calculator, and for the purpose of learning and understanding the nature of coding. It is also in terms of its functionality very convenient and practical and includes with it, history functionality, where all the outputs are collected and saved and can be called upon later.

This project also covers all parts pertaining to the course taken, meaning all topics covered in the class and homework, or otherwise, were covered.

Code Overview:

The code starts with the usual library calling, mainly <stdio.h> for our C functions, and <math.h> for all our mathematical functions.

Then it proceeds to name the functions, this approach was taken because it was intended that the main() function be up top and the other functions that are to be called are at the bottom, mainly for the sake of appearances.

This code also covers major topics taken in this course, mainly including the likes of structs, files, arrays, and memory allocation. This code consists of an extensive implementation of all terms and concepts clarified by the instructor, with the added benefit of being extremely user friendly and from the moment of running will also be very self-explanatory. For an in-depth manual, refer bellow.

User manual:

The program when run the first time will display something like this:Text

Description automatically generated

Displaying all the possible capabilities of the program. Any input by the user will be taken and an action will be performed.

Note: Do not enter a character. This will break the code and isn’t really the function of this calculator.

If any number that is above 12 and bellow 1 is entered, the program detects an error and re prompts the user to enter their desired function again, there are only 3 attempts so after the third wrong entry the program ends.

If 1 is entered, the user is prompted to enter an operator then they are requested to enter two values and the program prints the output.

Graphical user interface, text, application

Description automatically generated

If 2 is entered, the function allows calculation of any number with any exponent and can also calculate root of any number.

Text

Description automatically generated Text

Description automatically generated

If 3 is entered, the program can calculate the binary, hexadecimal and octal of a given integer.

Text

Description automatically generated

If 4 is entered, the program can find the values of X and Y, from simultaneous equations given that the values are a, b, c, I, j and k, from ax + by = c and ix + jy = k, and a, b and c from ax^2 + bx + c.

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

If 5 is entered, the number “a” taken from the user will be used in almost all the trigonometry functions.

Example:

Sin(a) cos(a) tan(a) arcsin(a) arccos(a) arctan(a) csc(a) sec(a) cot(a)

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

If 6 is entered, it calculates the output of a polynomial function and the output of its derivative as well.

Text

Description automatically generated

If 7 is entered, we have a Pythagorean theorem calculator, where we first enter the missing value, and then we enter the values of the other missing variables.

Hypotenuse (squared) = base (squared) + height (squared)

Text

Description automatically generated Text

Description automatically generated

If 8 is entered, we can find the largest or smallest integer, sorting, calculating average of all inputs in a set of integers of unknown number of values.

**This is where the concept of memory allocation was used, pointers was used for this.**

Text

Description automatically generatedText

Description automatically generated

If 9 is entered, the program calculates factorial, permutation, and combination of given parameters.

Text

Description automatically generatedText

Description automatically generated

If 10 is entered, the program calculates the distance between the reference point to all the provided points and presents the longest and shortest distances to the reference point.

**Structs was covered in this function.**

**[Unfortunately, this function was not working in Dev C++ but working perfectly in Visual Studio]**

Text

Description automatically generatedText

Description automatically generated

The numbers 11 and 12 are history managers, where 11 displays the history of the system calculator and 12 erases the calculator history.

**This part of the code deals entirely with files, continuously printing all outputs into a folder and can be called into the function any time.**

The openness of the calculator allows for more versatility in the future allowing for more functions to be added.

The code :::::::::

//Mohammad Rauf B2105.010002

//actively working calculator

#include<stdio.h>

#include<math.h>

#include<conio.h>

#include<windows.h>

#include <time.h>

void function1(); //Function to be called, Function defination bellow, leaving main up top.

void function2(); //Function to be called, Function defination bellow, leaving main up top.

void function3\_hexadecimal(int decimal); //Function to be called, Function defination bellow, leaving main up top.

void function3\_octal(int decimal); //Function to be called, Function defination bellow, leaving main up top.

void function3\_binary(int decimal); //Function to be called, Function defination bellow, leaving main up top.

void function4(); //Function to be called, Function defination bellow, leaving main up top.

void function5(); //Function to be called, Function defination bellow, leaving main up top.

void function6(); //Function to be called, Function defination bellow, leaving main up top.

void function7(); //Function to be called, Function defination bellow, leaving main up top.

void function8(); //Function to be called, Function defination bellow, leaving main up top.

void function9(); //Function to be called, Function defination bellow, leaving main up top.

void function10(); //Function to be called, Function defination bellow, leaving main up top.

double coordinates\_distance(double x, double y, double x2, double y2); //Function to be called, Function defination bellow, leaving main up top.

long factorial(double num); //Function to be called, Function defination bellow, leaving main up top.

void history\_logger(); //Function to be called, Function defination bellow, leaving main up top.

void history\_clear(); //Function to be called, Function defination bellow, leaving main up top.

int attempt = 3;

int main() { //main function initiation.

time\_t t = time(NULL);

struct tm\* tm = localtime(&t);

system("COLOR 03"); //output color set.

int function; //variable declaration.

char repeat; //charecter variavle declaration.

FILE\* history; //Folder address assignment.

history = fopen("History\_Depo.txt", "a"); //opening folder. "a" means open and update.

printf("\t B2105.010002\n");

printf("\t Mohammad Raufs Calculator\n");

printf("\t Welcome to my active working calculator with extra functions\n\n");

printf("\t Currently active functions are as follows\n\n\n");

printf("\tEnter 1, for common functions such as addition, subtraction, multiplication and division.\n\n");

printf("\tEnter 2, for power of and square root functions.\n\n");

printf("\tEnter 3, for finding the binary, octal, hexadecimal of a given decimal number\n\n");

printf("\tEnter 4, for Simultanious ax + by = c and quadratic equations equations ax^2 + bx + c.\n\n");

printf("\tEnter 5, for scientific calculator functions (sin, cosin, tan, arcsin, arccos, arctan, sec, cosec, cot).\n\n");

printf("\tEnter 6, for differential equation or integration functions.\n\n");

printf("\tEnter 7, for pythagorian theorem calculator. \n\n");

printf("\tEnter 8, for finding the largest or smallest integer, sorting, calculating average of \n\t\tall inputs in a set of integers of unknown no. of values.\n\n");

printf("\tEnter 9, for factorial permutation, combination.\n\n");

printf("\tEnter 10, for Coordinates corilations and between given number of points.\n\n");

printf("\tEnter 11, for History.\n\n\tEnter 12, for clearing history.\n\n");

printf("\nEnter the number of the function that you would like to use: \nFunction: ");

scanf("%d", &function); //scan user input.

fprintf(history, "%s\n", asctime(tm)); //prints in the folder

fclose(history); //closinf folder.

switch (function) { //checks user entry and performs whatever task accordingly.

case(1): {

system("cls"); system("COLOR 02"); //output color set.

function1(); //function called.

break;

}

case(2): {

system("cls"); system("COLOR 02"); //output color set.

function2(); //function called.

break;

}

case(3): {

system("cls"); system("COLOR 06"); //output color set.

int decimal; //function called.

printf("Enter a decimal number: ");

scanf("%d", &decimal);

function3\_hexadecimal(decimal);

function3\_octal(decimal);

function3\_binary(decimal);

printf("\n");

break;

}

case(4): {

system("cls"); system("COLOR 06"); //output color set.

function4(); //function called.

break;

}

case(5): {

system("cls"); system("COLOR 06"); //output color set.

function5(); //function called.

break;

}

case(6): {

system("cls"); system("COLOR 06"); //output color set.

function6(); //function called.

break;

}

case(7): {

system("cls"); system("COLOR 06"); //output color set.

function7(); //function called.

break;

}

case(8): {

system("cls"); system("COLOR 04"); //output color set.

function8(); //function called.

break;

}

case(9): {

system("cls"); system("COLOR 04"); //output color set.

function9(); //function called.

break;

}

case(10): {

system("cls"); system("COLOR 04"); //output color set.

function10(); //function called.

break;

}

case(11): {

system("cls"); system("COLOR F0"); //output color set.

history\_logger(); //function called.

break;

}

case(12): {

system("cls"); system("COLOR 0F"); //output color set.

history\_clear(); //function called.

break;

}

default: {

printf("Please make a valid entery \n");

attempt--; //decrement attempts by 1 units.

if (attempt != 0) {

system("cls");

printf("\n\n attempt remaining : %d\n", attempt);

return main();

}

else {

printf("\n No more attempts remaining.\n");

return 0;

break;

}

}

}

history = fopen("History\_Depo.txt", "a"); //folder address assigned to "history" variable.

fprintf(history, "\n\n");

fclose(history); //closing folder.

printf("continue? (y/n) : ");

getchar(); //Used this to stop a bug by the compiler.

scanf("%c", &repeat);

if (repeat == 'y') {

system("cls");

return main();

}

printf(" Have yourself a good day.\n\n\n");

return 0;

}

void function1() { //first function declared.

float num1, num2; //input variables declared.

char op; //charenter variable declared.

float result; //output variable declared.

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens folder.

printf("input an operator '+' , '-' , '/' , '\*' : ");

getchar(); //used to prevent a bug by the compiler.

scanf("%c", &op);

printf("enter 2 values:\n");

printf("value 1: "); scanf("%f", &num1);

printf("value 2: "); scanf("%f", &num2);

if (op == '+') {

result = num1 + num2; //addition.

}

else if (op == '-') {

result = num1 - num2; //subtraction.

}

else if (op == '\*') {

result = num1 \* num2; //multiplication.

}

else if (op == '/') {

result = num1 / num2; //division.

}

else {

printf("please make a valid entry");

fprintf(history, "Invalid data entry\n");

}

printf("The result of %.2f %c %.2f = %.3f \n\n", num1, op, num2, result);

fprintf(history ,"The result of %.2f %c %.2f = %.3f \n", num1, op, num2, result);

fclose(history); //closes folder.

}

void function2() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a");

int n;

printf("Enter: \n1. for Power function.\n2. for Square Root function.\nFunction: ");

scanf("%d", &n);

switch (n) {

case (1): {

double num1;

int power;

double result;

printf("enter a number: ");

scanf("%lf", &num1);

printf("enter a power: ");

scanf("%d", &power);

result = pow(num1, power); //using math.h library.

printf("The %d power of %.2f is equal to %.2f \n", power, num1, result); // prints result on screen.

fprintf(history, "The %d power of %.2f is equal to %.2f \n", power, num1, result); //prints output in history.

break;

}

case(2): {

double num1;

double result;

printf("enter a number: ");

scanf("%lf", &num1);

result = sqrt(num1); //using math.h library.

printf("The square root of %.2f is equal to %.2f \n", num1, result);

fprintf(history, "The square root of %.2f is equal to %.2f \n", num1, result);

break;

}

default: {

printf("\nfalse entry.\n");

fprintf(history, "\nfalse entry.\n");

break;

}

}

fclose(history); //closing folder.

}

void function3\_hexadecimal(int decimal) {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a");

int hexadecimal=decimal;

int hex\_arr[20]; //Set up an array for hexadecimal values.

int hex\_arr2[20]; //second array for reversal, msb and lsb.

int hex\_count; //counts total values.

int hex\_remainder; //the hexadecimal input.

int i = 0; //control variable.

printf("hexadecimal = ");

fprintf(history, "hexadecimal = "); //prints in file.

while (hexadecimal != 0) {

hex\_remainder = hexadecimal % 16; //takes remainder.

hexadecimal /= 16; //devider number and takes whole number.

hex\_arr[i] = hex\_remainder; //saves the remainders to the array.

i++; //increment the control by 1 every loop.

}

hex\_count = i; //saves the number of loops to the a variable to use as main control in out for loops.

for (i = 0; i < hex\_count; i++) {

hex\_arr2[i] = hex\_arr[hex\_count - 1 - i]; //reverses the array for Msb and Lsb.

}

for (i = 0; i < hex\_count; i++) {

switch (hex\_arr2[i]) {

case(10): printf("A"); fprintf(history, "A"); break; //hexadecimals rules and prints in file.

case(11): printf("B"); fprintf(history, "B"); break; //hexadecimals rules and prints in file.

case(12): printf("C"); fprintf(history, "C"); break; //hexadecimals rules and prints in file.

case(13): printf("D"); fprintf(history, "D"); break; //hexadecimals rules and prints in file.

case(14): printf("E"); fprintf(history, "E"); break; //hexadecimals rules and prints in file.

case(15): printf("F"); fprintf(history, "F"); break; //hexadecimals rules and prints in file.

default: printf("%d", hex\_arr2[i]); fprintf(history, "%d", hex\_arr2[i]); //hexadecimals rules and prints in file.

}

}

printf("\n");

fprintf(history, "\n");

fclose(history); //closing folder.

}

void function3\_octal(int decimal) {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

int octalNumber = 0; //initializing because itll be used in a loop.

int i = 1; //i should be 1 because anything multiplied with a 0 will result in a 0.

while (decimal != 0) //untill this condition (decimal != 0) is satesfied, it continues to loop.

{

octalNumber += (decimal % 8) \* i; //adds the remainder to the decimal place of i.

decimal /= 8; //number devided by 8 and whole number of result taken.

i \*= 10; //position increased by 1 digit.

}

printf("octal = %d\n", octalNumber);

fprintf(history, "octal = %d\n", octalNumber);

fclose(history); //closing folder.

}

void function3\_binary(int decimal) {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

int binarynumber[100]; //Similar concept to hexadecimals. refer to it.

int i = 1; //Similar concept to hexadecimals. refer to it.

int remainder; //Similar concept to hexadecimals. refer to it.

int count; //Similar concept to hexadecimals. refer to it.

printf("binary = "); //Similar concept to hexadecimals. refer to it.

fprintf(history, "binary = "); //Similar concept to hexadecimals. refer to it.

while (decimal != 0) { //Similar concept to hexadecimals. refer to it.

remainder = decimal % 2; //Similar concept to hexadecimals. refer to it.

binarynumber[i] = remainder; //Similar concept to hexadecimals. refer to it.

decimal /= 2; //Similar concept to hexadecimals. refer to it.

i++; //Similar concept to hexadecimals. refer to it.

}

count = i; //Similar concept to hexadecimals. refer to it.

int reversebin[100]; //Similar concept to hexadecimals. refer to it.

for (i = 0; i < count; i++) { //Similar concept to hexadecimals. refer to it.

reversebin[i] = binarynumber[count - 1 - i]; //Similar concept to hexadecimals. refer to it.

}

for (i = 0; i < count-1; i++) { //Similar concept to hexadecimals. refer to it.

printf("%d", reversebin[i]); //Similar concept to hexadecimals. refer to it.

fprintf(history, "%d", reversebin[i]); //Similar concept to hexadecimals. refer to it.

}

printf("\n");

fprintf(history, "\n");

fclose(history); //closing folder.

}

void function4() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

int n;

printf("Enter 1 for simultanious equations. \nEnter 2 for quadratic equations.\nFunction:");

scanf("%d", &n);

switch (n) {

case(1): {

double a, b, c; //declaration of first function variables.

double i, j, k; //declaration of first function variables.

double x, y; //declaration of x and y variables.

printf("Enter the values of coefficents A, B and C, of the first equation of the form Ax+By=C\n");

printf("Enter A : "); scanf("%lf", &a);

printf("Enter B : "); scanf("%lf", &b);

printf("Enter C : "); scanf("%lf", &c);

printf("Enter the values of coefficents I, J and K, of the second equation of the form Ix+Jy=K\n");

printf("Enter I : "); scanf("%lf", &i);

printf("Enter J : "); scanf("%lf", &j);

printf("Enter K : "); scanf("%lf", &k);

fprintf(history, "Simultanious equations:\na=%f\nb=%f\nc=%f\n\ni=%f\nj=%f\nk=%f\n", a, b, c, i, j, k); //prints input variables in folder.

if (((a \* j - i \* b) != 0) && ((b \* i - j \* a) != 0)){ //checks if the equation is undefined.

printf("The solution to the equations is unique\n");

fprintf(history, "The solution to the equations is unique\n");

x = (c \* j - k \* b) / (a \* j - i \* b); //the equation moulded to work in any instance the condition is satesfied.

y = (c \* i - k \* a) / (b \* i - j \* a); //the equation moulded to work in any instance the condition is satesfied.

printf("The value of x=%.3f\n", x);

fprintf(history, "The value of x=%.3f\n", x); //prints the result is history.

printf("The value of y=%.3f\n\n", y);

fprintf(history, "The value of y=%.3f\n\n", y); //prints the result is history.

}

else if (((a \* j - i \* b) == 0) && ((b \* i - j \* a) == 0) && ((c \* j - k \* b) == 0) && ((c \* i - k \* a) == 0)){ //checks if the equation is undefined.

printf("Infinitely many solutions are piossible\n");

fprintf(history, "Infinitely many solutions are piossible\n"); //prints the result is history.

printf("The value of x can be varied and y can be calculated according to x's value using relation\n");

fprintf(history, "The value of x can be varied and y can be calculated according to x's value using relation\n"); //prints the result is history.

printf("y=%.3f+(%.3f)x\n\n", (c / b), (-1 \* a / b));

fprintf(history, "y=%.3f+(%.3f)x\n\n", (c / b), (-1 \* a / b)); //prints the result is history.

}

else if (((a \* j - i \* b) == 0) && ((b \* i - j \* a) == 0) && ((c \* j - k \* b) != 0) && ((c \* i - k \* a) != 0)){ //checks if the equation is undefined.

printf("No possible solution\n\n\n");

fprintf(history, "No possible solution\n\n\n"); //prints the result is history.

break;

}

break;

}

case(2): {

double a, b, c;

double root\_check, root1, root2;

double real\_root, imagenary\_root;

printf("Enter coefficients A, B and C: \n");

printf("Enter A : "); scanf("%lf", &a);

printf("Enter B : "); scanf("%lf", &b);

printf("Enter C : "); scanf("%lf", &c);

fprintf(history, "Quadratic Equations:\nA=%f\nB=%f\nC=%f\n", a, b, c);

root\_check = b \* b - 4 \* a \* c; //checks for roots.

if (root\_check > 0) { //if statement to test.

root1 = (-b + sqrt(root\_check)) / (2 \* a); //general quadratic formula for first root.

root2 = (-b - sqrt(root\_check)) / (2 \* a); //general quadratic formula for second root.

printf("root 1 = %.3f \n", root1);

printf("root 2 = %.3f \n\n", root2);

fprintf(history, "root 1 = %.3f \n", root1); //prints result in file.

fprintf(history, "root 2 = %.3f \n\n", root2); //prints result in file.

}

else if (root\_check == 0) {

root1 = root2 = -b / (2 \* a); //general quadratic formula for second root.

printf("x 1 and x 2 is equal to %.3f \n\n", root1);

fprintf(history, "x 1 and x 2 is equal to %.3f \n\n", root1); //prints result in file.

}

else { //imaginary case.

real\_root = -b / (2 \* a); //finds the root part that is calsulatable.

imagenary\_root = sqrt(-root\_check) / (2 \* a); //finds the soefficient of the imaginary part.

printf("first root = %.2lf+%.2lfi and second root = %.2f-%.2fi \n\n", real\_root, imagenary\_root, real\_root, imagenary\_root);

fprintf(history, "first root = %.2lf+%.2lfi and second root = %.2f-%.2fi \n\n", real\_root, imagenary\_root, real\_root, imagenary\_root); //prints result in file.

}

break;

}

default: printf("invalid entry."); fprintf(history, "invalid entry.");

break;

}

fclose(history); //closing folder.

}

void function5() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

int n;

double num, radians, degrees;

printf("Degrees or Radians: \nEnter 1 for degrees.\nEnter 2 for radians.\nNumber:");

scanf("%d", &n);

printf("Enter your number: ");

scanf("%lf", &degrees);

switch (n) {

case(1): {

radians= 0.0174532925 \* degrees;

printf("\n\nsin = %.3f\ncosin = %.3f\ntan = %.3f\narcsin = %.3f\narccos = %.3f\narctan = %.3f\nsec = %.3f\ncosec = %.3f\ncot = %.3f\n\n", sin(radians), cos(radians), tan(radians), asin(radians), acos(radians), atan(radians), 1 / cos(radians), 1 / sin(radians), 1 / tan(radians));

fprintf(history, "\n\nsin = %.3f\ncosin = %.3f\ntan = %.3f\narcsin = %.3f\narccos = %.3f\narctan = %.3f\nsec = %.3f\ncosec = %.3f\ncot = %.3f\n\n", sin(radians), cos(radians), tan(radians), asin(radians), acos(radians), atan(radians), 1 / cos(radians), 1 / sin(radians), 1 / tan(radians));

break;

}

case(2): {

radians = degrees;

printf("\n\nsin = %.3f\ncosin = %.3f\ntan = %.3f\narcsin = %.3f\narccos = %.3f\narctan = %.3f\nsec = %.3f\ncosec = %.3f\ncot = %.3f\n\n", sin(radians), cos(radians), tan(radians), asin(radians), acos(radians), atan(radians), 1 / cos(radians), 1 / sin(radians), 1 / tan(radians));

fprintf(history, "\n\nsin = %.3f\ncosin = %.3f\ntan = %.3f\narcsin = %.3f\narccos = %.3f\narctan = %.3f\nsec = %.3f\ncosec = %.3f\ncot = %.3f\n\n", sin(radians), cos(radians), tan(radians), asin(radians), acos(radians), atan(radians), 1 / cos(radians), 1 / sin(radians), 1 / tan(radians));

break;

}

default: {printf("invalid entry.\n\n\n"); fprintf(history, "invalid entry.\n\n\n"); }

}

fclose(history); //closing folder.

}

/\*Function above basically follows the same pattern of the others taking 1 value predetermined if radians or degrees(which then converted in radians) using math.h library to find all of its trig function results.\*/

void function6() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

float x, a[10], y1, dy1, p, d[10], pd = 0, ps;

int deg, i;

printf("Enter the degree of polynomial equation: "); //askes user for degree of polynomial.

scanf("%d", &deg);

fprintf(history, "degree of polynomial equation = %d", deg); //prints the value inside of history.

printf("Enter the value of x for which the equation is to be evaluated: ");

scanf("%f", &x);

fprintf(history, "value of X for which the equation is to be evaluated= %f", x);//prints the value inside of history.

for (i = 0; i <= deg; i++) {

printf("Enter the coefficient of x to the power %d: ", i);

scanf("%f", &a[i]);

fprintf(history, "coefficient of x to the power of %d is %f", i, a[i]); //prints the value inside of history.

}

p = a[deg];

for (i = deg; i >= 1; i--) {

p = (a[i - 1] + x \* p);

}

y1 = p;

for (i = 0; i <= deg; i++) {

ps = pow(x, deg - (i + 1));

d[i] = (deg - i) \* a[deg - i] \* ps;

pd = pd + d[i];

}

dy1 = pd;

printf("\n\nThe value of polynomial equation for the value of x = %.2f is: %.2f", x, y1);

printf("\nThe value of the derivative of the polynomial equation at x = %.2f is: %.2f\n\n", x, dy1);

fprintf(history, "\n\nThe value of polynomial equation for the value of x = %.2f is: %.2f", x, y1);

fprintf(history, "\nThe value of the derivative of the polynomial equation at x = %.2f is: %.2f\n\n", x, dy1);

fclose(history); //closing folder.

}

void function7() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

int n;

double hypotinuse, base, hight;

printf("\n\n which missing value is needed to be calculated.\n\tEnter 1. Hypotinuse\n\tEnter 2. Base\n\tEnter 3. Hight\nSelection:");

scanf("%d", &n);

printf("Enter your Values:\n\n");

switch (n) {

case(1): {

printf("\nbase = "); scanf("%lf", &base); fprintf(history,"\nbase = %.3f", base); //takes base input from user and prints it in history.

printf("height = "); scanf("%lf", &hight); fprintf(history, "\nhight = %.3f", hight); //takes hight input from user and prints it in history.

hypotinuse = sqrt(pow(base, 2) + pow(hight, 2)); //calculates hypotinues from the other two variables.

printf("\nhypotinuse = %.3f\n\n", hypotinuse);

fprintf(history, "\nhypotinuse = %.3f\n\n", hypotinuse);

break;

}

case(2): {

printf("height = "); scanf("%lf", &hight); fprintf(history, "\nhight = %.3f", hight); //takes hight input from user and prints it in history.

printf("\nhypotinuse = "); scanf("%lf", &hypotinuse); fprintf(history, "\nhypotinuse = %.3f", hypotinuse); //takes hypotinuse input from user and prints it in history.

base = sqrt(pow(hypotinuse, 2) - pow(hight, 2)); //calculates base from the other two variables.

printf("\nbase = %.3f\n\n", base);

fprintf(history, "\nbase = %.3f\n\n", base);

break;

}

case(3): {

printf("\nbase = "); scanf("%lf", &base); fprintf(history, "\nbase = %.3f", base); //takes base input from user and prints it in history.

printf("hypotinuse = "); scanf("%lf", &hypotinuse); fprintf(history, "\nhypotinuse = %.3f", hypotinuse); //takes hypotinuse input from user and prints it in history.

hight = sqrt(pow(hypotinuse, 2) - pow(base, 2)); //calculates Height from the other two variables.

printf("\nhight = %.3f\n\n", hight);

fprintf(history, "\nheight = %.3f\n\n", hight);

break;

}

default: {printf("\n\nInvalid entry\n\n"); fprintf(history, "\n\nInvalid entry\n\n"); break; }

}

fclose(history); //closing folder.

}

void function8() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

double\* ptr;

int n;

double temp;

int i;

int j;

printf("\n\nEnter the number of values: ");

scanf("%d", &n);

ptr = (double\*)malloc(n \* sizeof(double)); //memory allocation of an array.

for (i = 0; i < n; i++) { //takes number of inputs from the user.

printf("entry %d : ", i + 1);

scanf("%lf", &ptr[i]);

fprintf(history, "entry %d = %.3f\n", i + 1, ptr[i]); //prints to history.

}

for (i = 0; i < n; i++) { //ascending sorter.

for (j = i + 1; j < n; j++) {

if (ptr[i] > ptr[j]) {

temp = ptr[i];

ptr[i] = ptr[j];

ptr[j] = temp;

}

}

}

printf("Ascending order = ");

fprintf(history, "Ascending order = ");

for (i = 0; i < n; i++) {

printf("%.2f\t", ptr[i]);

fprintf(history, "%.2f\t", ptr[i]);

}

printf("\n"); fprintf(history, "\n");

for (i = 0; i < n; i++) { //descending sorter.

for (j = i + 1; j < n; j++) {

if (ptr[i] < ptr[j]) {

temp = ptr[i];

ptr[i] = ptr[j];

ptr[j] = temp;

}

}

}

printf("Descending order = "); fprintf(history, "Descending order = ");

for (i = 0; i < n; i++) {

printf("%.2f\t", ptr[i]); fprintf(history, "%.2f\t", ptr[i]);

}

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

printf("Largest Value is %.2f\nSmallest Value is %.2f\n", ptr[0], ptr[n - 1]);

fprintf(history, "Largest Value is %.2f\nSmallest Value is %.2f\n", ptr[0], ptr[n - 1]);

double average, sum = 0;

for (i = 0; i < n; i++) {

sum += (double)ptr[i];

}

printf("The sum of the values = %.3f\nThe average of provided numbers = %.3f\n\n\n\n", sum, sum / n);

fprintf(history, "The sum of the values = %.3f\nThe average of provided numbers = %.3f\n\n\n\n", sum, sum / n);

fclose(history); //closing folder.

}

void function9() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

long fact, combination, permutation;

int r, n;

char selection;

printf("\nEnter A for Factorial.\nEnter B for permutation.\nEnter C for combination.\n ");

printf("TYPE IN X FOR ALL : ");

getchar();

scanf("%c", &selection);

switch (selection) {

case('A'):

case('a'): {

printf("Enter r:"); scanf("%d", &r); fprintf(history, "r = %d\n", r);

fact = factorial(r);

printf("\nFactorial of %d (%d!) is equal to %d\n\n", r, r, fact);

fprintf(history, "\nFactorial of %d (%d!) is equal to %d\n\n", r, r, fact);

break;

}

case('B'):

case('b') : {

printf("Enter n:"); scanf("%d", &n); fprintf(history, "n = %d\n", n);

printf("Enter r:"); scanf("%d", &r); fprintf(history, "r = %d\n", r);

permutation = factorial(n) / factorial(n - r);

printf("\nThe value of %dP%d is equal to %d\n\n", n,r,permutation);

fprintf(history, "\nThe value of %dP%d is equal to %d\n\n", n, r, permutation);

break;

}

case('C'):

case('c') : {

printf("Enter n:"); scanf("%d", &n); fprintf(history, "n = %d\n", n);

printf("Enter r:"); scanf("%d", &r); fprintf(history, "r = %d\n", r);

permutation = factorial(n) / factorial(n - r);

combination = factorial(r);

printf("\nThe value of %dC%d is equal to %d\n\n", n, r, permutation/combination);

fprintf(history, "\nThe value of %dC%d is equal to %d\n\n", n, r, permutation / combination);

break;

}

case('x'): {

printf("Enter n:"); scanf("%d", &n);

printf("Enter r:"); scanf("%d", &r);

permutation = factorial(n) / factorial(n - r);

combination = factorial(r);

printf("\n%d! = %d", r, combination);

printf("\n%dP%d = %d", n, r, permutation);

printf("\n%dC%d = %d\n\n", n, r, permutation / combination);

fprintf(history, "\n%d! = %d", r, combination);

fprintf(history, "\n%dP%d = %d", n, r, permutation);

fprintf(history, "\n%dC%d = %d\n\n", n, r, permutation / combination);

break;

}

default: {printf("\nFalse entry.\n"); fprintf(history, "\nFalse entry.\n"); }

}

fclose(history); //closing folder.

}

long factorial(double num) {

long long factorial = 1;

while (num > 0) {

factorial \*= num;

num--;

}

return factorial;

}

double coordinates\_distance(double x, double y, double x2, double y2) {

double distance, xr, yr;

xr = pow(x2 - x, 2);

yr = pow(y2 - y, 2);

distance = sqrt(xr + yr);

return distance;

}

void function10() {

FILE\* history; //folder address assigned to "history" variable.

history = fopen("History\_Depo.txt", "a"); //opens and continues input. but doesnt change already available contents.

struct coordinates {

double x;

double y;

};

struct coordinates point[100], referance\_point;

long double distance[100], control[100], control\_2[100];

int n;

int i;

double a, b, c;

printf("Enter number of coordinates: "); scanf("%d", &n);

printf("\nEnter your coordinates: \n");

for (i = 0; i < n; i++) {

printf("input x%d coordinate: ", i + 1); scanf("%lf", &point[i].x);

fprintf(history, "input x%d coordinate = %.3f\n", i + 1, point[i].x); //prints values to history file.

printf("input y%d coordinate: ", i + 1); scanf("%lf", &point[i].y);

fprintf(history, "input y%d coordinate = %.3f\n", i + 1, point[i].y); //prints values to history file.

}

printf("input referance coordinate X: "); scanf("%lf", &referance\_point.x);

printf("input referance coordinate Y: "); scanf("%lf", &referance\_point.y);

fprintf(history, "Referance coordinate X= %.3f\n", referance\_point.x); //prints values to history file.

fprintf(history, "Referance coordinate Y= %.3f\n", referance\_point.y); //prints values to history file.

for (i = 0; i < n; i++) {

distance[i] = coordinates\_distance(referance\_point.x, referance\_point.y, point[i].x, point[i].y);

}

for (i = 0; i < n; i++) {

a = distance[i];

printf("Distance is %.3f units between points (%.1f, %.1f) and points (%.1f, %.1f).\n", a, referance\_point.x, referance\_point.y, point[i].x, point[i].y);

fprintf(history, "Distance is %.3f units between points (%.1f, %.1f) and points (%.1f, %.1f).\n", a, referance\_point.x, referance\_point.y, point[i].x, point[i].y);

control[i] = distance[i];

control\_2[i] = distance[i];

}

for (i = 0; i < n; i++) {

if (control[0] < control[i]) {

control[0] = control[i];

}

}

printf("\n");

for (i = 0; i < n; i++) {

if (control[0] == distance[i]) {

b = distance[i];

printf("LARGEST Distance is %.3f units between points (%.1f, %.1f) and points (%.1f, %.1f).\n", b, referance\_point.x, referance\_point.y, point[i].x, point[i].y);

fprintf(history, "LARGEST Distance is %.3f units between points (%.1f, %.1f) and points (%.1f, %.1f).\n", b, referance\_point.x, referance\_point.y, point[i].x, point[i].y);

}

}

for (i = 1; i < n; i++) {

if (control\_2[0] > control\_2[i]) {

control\_2[0] = control\_2[i];

}

}

printf("\n"); fprintf(history, "\n");

for (i = 0; i < n; i++) {

if (control\_2[0] == distance[i]) {

c = distance[i];

printf("SMALLEST Distance is %.3f units between points (%.1f, %.1f) and points (%.1f, %.1f).\n", c, referance\_point.x, referance\_point.y, point[i].x, point[i].y);

fprintf(history, "SMALLEST Distance is %.3f units between points (%.1f, %.1f) and points (%.1f, %.1f).\n", c, referance\_point.x, referance\_point.y, point[i].x, point[i].y);

}

}

printf("\n"); fprintf(history, "\n");

fclose(history); //closing folder.

}

void history\_logger() {

FILE\* history; //folder address assigned to "history" variable.

char filename[100000], c;

history = fopen("History\_Depo.txt", "r"); //opens and continues input. but doesnt change already available contents.

if (history == NULL)

{

printf("Cannot open file \n");

exit(0);

}

c = fgetc(history);

while (c != EOF)

{

printf("%c", c);

c = fgetc(history);

}

fclose(history); //closing folder.

}

void history\_clear() {

FILE\* history; //folder address assigned to "history" variable.

char filename[100000], c;

history = fopen("History\_Depo.txt", "w"); //opens and reads folder.

fclose(history); //closing folder.

}