**Daniyal Khan**

**3765942**

**CS-2263**

# Assignment 4

## Source code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

float ** dataReadIn(char *fileName, int *numRows);
float euclideanDist(float * row1, float * row2, int length);

#define NUM_OF_MEASUREMENTS 4

int main(int argc, char **argv) {
    if (argc < 2) {
        printf("Usage: %s <filename>.txt\n", argv[0]);
        return 1;
    }
    int numRows = 0;
    float **patientData = dataReadIn(argv[1], &numRows);
    if (patientData == NULL) {
        return 1;
    }

    float p1, m1, m2, m3, m4;
    printf("Input patient data: ");
    scanf("%f, %f, %f, %f, %f", &p1, &m1, &m2, &m3, &m4);
    float patientToBeDiagnosed[] = {p1, m1, m2, m3, m4};

    float dist;
    int closetPatientNum = 0;
    float smallestDist = euclideanDist(patientData[0], patientToBeDiagnosed,
NUM_OF_MEASUREMENTS);
```

```
    for (int i = 0; i < numRows; i++) {
        dist = euclideanDist(patientData[i], patientToBeDiagnosed, NUM_OF_MEASUREMENTS);
        if (dist < smallestDist) {
            closetPatientNum = i;
            smallestDist = dist;
        }
    }

    printf("Diagnosis of the closest previous patient: %.0f \n", patientData[closetPatientNum][5]);

    for (int i = 0; i < numRows; i++) {
        free(patientData[i]);
    }
    free(patientData);

    return 0;
}

float euclideanDist(float * row1, float * row2, int length) {
    float sum = 0.0;
    for (int i = 1; i <= length; i++) { // skip the patient number
        float diff = row1[i] - row2[i];
        sum += diff * diff;
    }
    return sqrt(sum);
}

float ** dataReadIn(char * fileName, int * numRows) {
    FILE *fptr = fopen(fileName, "r");
    if (fptr == NULL) {
        printf("File not found!\n");
        return NULL;
```

```
}

int ch;
while ((ch = fgetc(fptr)) != EOF) { // count the number of rows
    if (ch == '\n') {
        (*numRows)++;
    }
}
if (*numRows == 0) {
    printf("Not enough data in file!\n");
    return NULL;
}

float ** data = malloc((*numRows) * sizeof(float*)); // allocate space of the heap for the 2D Array
if (data == NULL) {
    printf("Error allocating space!\n");
    return NULL;
}
for (int i = 0; i < *numRows; i++) {
    data[i] = malloc(6 * sizeof(float));
    if (data[i] == NULL) {
        printf("Error allocating space!\n");
        // freee all previously allocated rows
        for (int j = 0; j < i; j++) {
            free(data[j]);
        }
        free(data);
        return NULL;
    }
}
rewind(fptr); // reset file pointer to the beginning

for (int i = 0; i < *numRows; i++) {
```

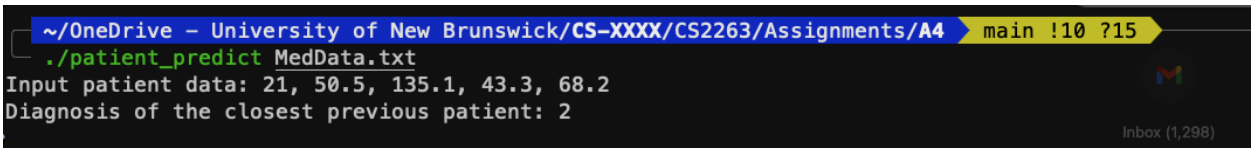
```
        fscanf(fptr, "%f,%f,%f,%f,%f,%f", &data[i][0], &data[i][1], &data[i][2], &data[i][3], &data[i][4],
&data[i][5]);
    }
    fclose(fptr);

    return data;
}
```

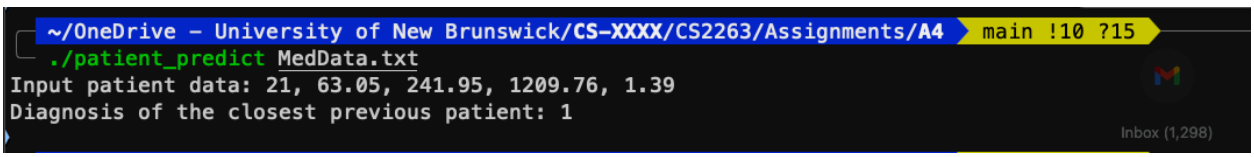
# Makefile:

```
GCC = gcc
CFLAGS = -g -Wall -Wshadow -lm

all: patient_predict

patient_predict: patient_predict.o
    $(GCC) $(CFLAGS) patient_predict.o -o patient_predict

patient_predict.o: patient_predict.c
    $(GCC) $(CFLAGS) -c patient_predict.c

test0: patient_predict
    ./patient_predict MedData.txt < Test/input0.txt > Test/output0.txt

test1: patient_predict
    ./patient_predict MedData.txt < Test/input1.txt > Test/output1.txt

test2: patient_predict
    ./patient_predict MedData.txt < Test/input2.txt > Test/output2.txt
```

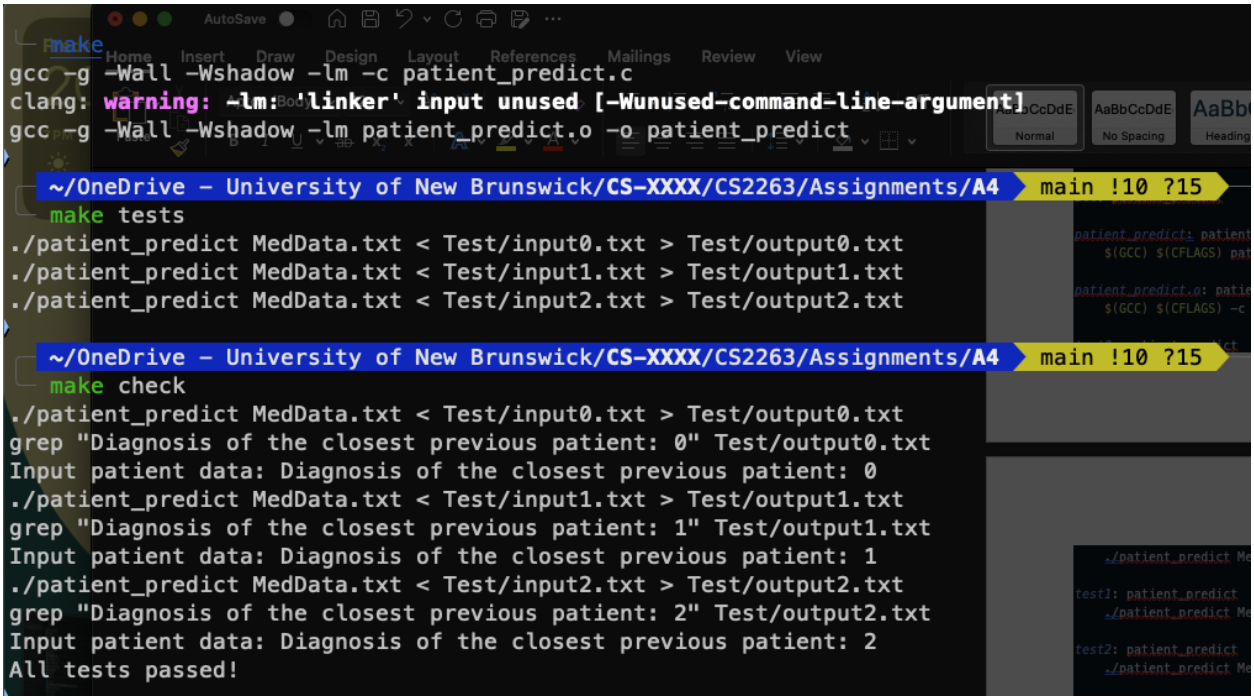
```
tests: test0 test1 test2

check0: test0
    grep "Diagnosis of the closest previous patient: 0" Test/output0.txt

check1: test1
    grep "Diagnosis of the closest previous patient: 1" Test/output1.txt

check2: test2
    grep "Diagnosis of the closest previous patient: 2" Test/output2.txt

check: check0 check1 check2
    @echo "All tests passed!"

clean:
    rm -f *.o patient_predict
```

# Testing:



Testing with MakeFile:




Inputs used:

1. 21, 58.01695695, 237.026522, 1185.13261, 1.706318785
2. 22, 63.04762625, 241.9527672, 1209.763836, 1.394672244
3. 23, 85.3639715, 190.24415, 951.2207501, 0.411893893