

## ESE 124 Fall 2023 Midterm

ID: 114992653

First name Last name Vinson Jin

82.5

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- a) Read Data (10 points)

Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.

- b) Store Data as 1D Array (10 points)

Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.

- c) Using Bubble Sort (10 points)

Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.

- d) Write the Solution to Output File (5 points)

Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."

Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- a) Define the variables and initial them (5 points).

- b) Find the next term related to the previous term (5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

#### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011,  $1+1+0+1+0=3$ ).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

Q1) #include <stdio.h> // all the libraries

#include <math.h>

#include <stdlib.h>

int main()

{ FILE \*fin1 \*fin2 \*fout;

fopen fin1("input1.txt", "r"); // opens fin1, fin2, fout

fopen fin2("input2.txt", "r");

fopen fout("output.txt", "w");

float arr1[INT\_MAX]; float arr2[INT\_MAX]

int i=0; int j=0; int k=0; int x=0; // initialize variables

while ((fscanf(fin1, "%d" &arr1[i])) != EOF) // scans fin1 and assigns to arr1

{ i++ }

while ((fscanf(fin2, "%d" &arr2[j])) != EOF) // scans fin2 and assigns to arr2

{ j++ }

float arr3[i+j]; // 3rd array has size of arr1+arr2

if ((fin1 == null) || (fin2 == null) || (fout == null)) // checks if output/input files are null

{ printf("cannot read file");

exit(1);

} int y=i+j; // initialize variable

for (k=0; k<(i+j); k++) // deleting duplicates

{ for (x=k+1; x<(i+j)-1; x++) // loops twice, first to compare the

{ if (arr3[k] == arr3[x]) // number with the rest, then if it finds duplicates shifts to front and

{ arr3[k] = arr3[x]; // deletes the back

y--;

}

}

}

Qc :- 8

-3

-5 duplicates Next Page

```

int temp; For bubble sort / swapping array values
for (i=0; i < (sizeof(arr3) / sizeof(arr3[0]))-1; i++)
{
    for (j=0; j < (sizeof(arr3) / sizeof(arr3[0]))-i-1; j++)
    // double for loop. The first i loop will compare the first element
    // of array with the rest then swap if necessary. Then it is
    // increased by 1 and thus the second element will be
    // compared with the rest of the array.
    {
        if (arr[j] > arr[j+1]) // compares to see if greater
        {
            temp = arr[j]; // Swaps them using temp
            arr[j] = arr[j+1]; // iterates this multiple times
            arr[j+1] = temp;
        }
    }
    fprintf(fout, "%s", "The Third array (result) is: "); // prints in output
    for (i=0; i < (sizeof(arr3) / sizeof(arr3[0])); i++) // iterates
    // through arr3
    {
        printf("%d", arr3[i]);
        fprintf(fout, "%d", arr3[i]); // prints out each element of arr3
        // in output file
    }
    float sum, average; // for calculating sum and average
    for (i=0; i < (sizeof(arr3) / sizeof(arr3[i])); i++) // adds up all elements
    // of arr3
    {
        sum += arr3[i];
    }
    average = sum / (sizeof(arr3) / sizeof(arr3[i])); // average is sum / length
    fprintf(fout, "%s %d\n %s %d", "The Sum is", sum, "The
    Average is", average); // prints out results in the output file
    fclose(fin1); // close all files
    fclose(fin2);
    fclose(fout);
    return 0;
}

```

Vinson Jin 1144926

Q2) #include <stdio.h> // libraries  
#include <math.h>

Vinson J. J  
11499265

```
int main() {
    float term = -1 // initializing variables
    float x; float smallsum;
    int n; int i // Scans x and n from keyboard
    float sum = 0; printf("Enter x and n"); scanf("%d %d", &x, &n);
    float arr[n];
    for (i = 1; i <= n; i++) {
        term = (x-1) // (x-1)
        term = power(term, i); // (x-1)^i ← i is always same as denominator terms
        term = (-1) * (term / i) // (x-1)^i / i
        sum += term; // -3
        arr[i-1] = term; // sets the first three terms as smallest and records indexes
    }
    bigsum = arr[0] + arr[1] + arr[2]; int index1 = 0; int index2 = 1; int index3 = 2;
    for (i = 0; i <= n-3; i++) {
        if (arr[i] + arr[i+1] + arr[i+2] < smallsum) { // -3
            // if three terms sum < smallest sum
            smallsum = arr[i] + arr[i+1] + arr[i+2]; // replaces smallsum
            index1 = i; // and indexes if true
            index2 = i+1;
            index3 = i+2;
        }
    }
    printf("%f", sum); // prints sum of n terms and consecutive term sums
    printf("Smallest sum of three terms and their indexes: ", smallsum, index1, index2, index3);
    return 0;
}
```

Q2: -6

Vinson Jin 114992653

```
#include <stdio.h> #define STR_LENGTH 32
```

```
#include <stdlib.h>
```

```
#include <string.h> //include libraries
```

```
int main()
```

```
{ FILE *fSen *fname *fout; //declare 3 files
```

```
here; char c; char d; //initialize characters
```

```
int true = 0;  
char variable [STR_LENGTH]; //declare a array of characters
```

```
fopen = fopen("input-sentence.txt", "r"); //open 3 files
```

```
fopen = fopen("names.txt", "r");
```

```
fopen = fopen("output-sentence.txt", "w");
```

```
if((fSen == null) || (fname == null) || (fout == null)) //check to make sure  
//they are not null
```

```
{ printf("file cannot open");
```

```
exit(1);
```

```
while ( (c = fgetc(fSen) != EOF) ) //gets the first letter of the sentence file  
{ d = fgetc(fname); //gets first letter of name file and assigns to d.
```

```
while (d != ",") //detects if characters are matching
```

```
{ if (c == d)
```

```
{ e = fgetc(fSen);
```

```
d = fgetc(fname);
```

```
true = 1
```

```
}  
else
```

```
{ true = 0  
break }
```

```
if (true)
```

```
fprintf(fout, "%c", toupper(c)); //if matches upper case that character
```

```
else  
fprintf(fout, "%c", c); } //else its the same.
```

Q4)

#include &lt;stdio.h&gt;

#include &lt;math.h&gt;

int main()

{ hex par, par1, par2;

printf("Enter hex");

scanf("%x", &amp;par);

par1 = par &amp; ~(7);

par2 = par &amp; (7);

+1.5

Q3 continued;

fclose(fsen);

fclose(fname);

fclose(fout);

return 0;





## ESE 124 Fall 2023 Midterm

ID: 114225490

First name Last name Jonathan Behge

66  
69.5

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

#### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

Q1:

#include &lt;stdio.h&gt;

#include &lt;math.h&gt;

int main()

FILE \*file1, \*file2;

file1 = fopen("input1.txt", "r"); // Reads "input1.txt" as file1

file2 = fopen("input2.txt", "w"); // Reads "input2.txt" as file2

float arr1[30]; // Make two float arrays because at most can be 30

float arr2[30]; float arr3[30]; // Third array used for storing arranged values

float sum = 0; // Sum of arranged numbers

float avg; // Average of arranged numbers

int i, j, count = 0; // Will be used to traverse through files/arrays

while(fscanf(file1, "%f", arr[i]) != EOF) // creates arr[1] and arr[2] and compare them to arr[3]

for(i=0; i &lt; sizeof(arr)/sizeof(int); i++)

if(fscanf(file2, "%f", arr2[j]) != arr[i]) // If number in arr1 isn't in arr2

arr3[count] = arr2[j]; count++;

else if(arr1[i] != arr1[j])

arr3[count] = arr1[j]; count++;

// If number in arr1 isn't the same in arr1.

for(i=0; i &lt; sizeof(arr3)/sizeof(int); i++) // Used to sort duplicates in arr3

for(j=0; j &lt; i; j++)

if(arr3[i] == arr3[j])

arr3[j] = 0; count--;

// Will use count to find avg.

Q1: -4.5

-0.5

-0.5

30 cannot store two arrays, doesn't it need to be 60?

Missing

b -&gt; Array pointer

Isn't it a float array?



```

float max = arr3[0];
int maxcount = 0; i = 1; end = 0;
while(1);

```

```

if(max > arr3[i])

```

```

    arr3[maxcount] = arr3[i];

```

```

    arr3[i] = max; maxcount++; i++;

```

```

else if(max < arr3[i])

```

```

    max = arr3[i];

```

```

    maxcount++;
    i++;

```

```

for(j = 0; j < sizeof(arr3); j++)

```

```

    if(arr3[j] < arr3[j+1])

```

```

        end++;

```

```

if(end == sizeof(arr3))

```

```

    break;

```

```

for(i = 0; i < sizeof(arr3); i++) // prints the numbers in array 3.

```

```

    printf("%d", arr3[i]);

```

```

    sum += arr3[i];

```

// sum of numbers in array

```

avg = sum / count;

```

```

printf("%d", "%d", sum, avg); // prints average and sum of array 3.

```

Q1

Output file? -2.5

// Will be used to sort having max;  
 // maxcount to determine where max is  
 // end to decide if the numbers are  
 in order

// Moves max term forward and  
 moves the previous term back

-1

// Changes max term to the  
 next term while not changing  
 order

// Checks if all the numbers are  
 in ascending order with an  
 if condition to match the  
 size of the array



$$\ln x = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \frac{(x-1)^5}{5} - \dots$$

$$\frac{(-1)^{n+1} (x-1)^{n+1}}{n+1} = \frac{(-1)^{n+1} (x-1)^{n+1}}{n+1} - \frac{n}{(-1)^{n+1} (x-1)^{n+1}} = \frac{(-1)^{n+1} (x-1)^n}{n+1}$$

#include <stdio.h>

#include <math.h>

int main() {

int count = 1;

float x, first = 0, second = 0, third = 0, consec sum = 0; // variables

float sum = 0;

float temp;

int n;

printf("value x: and value n");

scanf("%f, %d", &x, &n);

getchar();

do {

temp = temp \*  $\frac{(-1)^{\text{count}} (x-1)^{\text{count}}}{\text{count}+1}$ ; -1

sum += temp;

if (count == 1)

// Case to get consec term 1

{

first = temp;

-5

else if (count == 2)

// Case to get consec term 2

{

second = temp;

}

else if (count == 3)

// Case to get consec term 3

{

third = temp;

}

if (count > 3)

// More terms smaller and changes sum if series keeps going

{

first = second;

second = third;

third = temp;

consec sum = first + second + third;

// consecutive sum

```

} while (count < n)
printf("%f", sum);
printf("%f", average sum);
printf("%f %f %f", first, second, third); // three consecutive binary
}

```

// result of Taylor series  
 // consecutive sum  
 // three consecutive binary

Q3

```

#include <stdio.h>
#include <stdlib.h>
int main() {

```

```

    int i=0, j=0, account=0;
    FILE *f;
    sentence = fopen("input_sentence.txt", "r"); // sentence is input sentence
    name = fopen("names.txt", "w"); // variables to go through loops and keep track of array position
    char sent_string[100];
    char name_string[100]; int arr[100];
    while (!feof(sentence) && sent_string[i] != '\0') // sentence is input sentence
    {
        fgets(sent_string, 100, sentence); // Makes all the strings in sentence file into sent_string
        for (i=0; i < strlen(sent_string); i++) // Loop through sent_string
        {
            if (fscanf(name, "%s", &name_string[i]) == sent_string[i]) // Checks if the strings are
            {
                arr[account] = i;
                account++;
            }
            j++;
        }
        account = 0;
        for (i=0; i < strlen(sent_string); i++) // Go through array to check if any matches input number
        {
            if (arr[account] == i) // If match uppercase the position character i which was saved
            {
                fprintf(f, "%s", sent_string[i]); // print out the strings in sent_string.
            }
        }
    }
}

```

Q3:-20

-10

-2.5

-2.5 - 5 (r, w)



## ESE 124 Fall 2023 Midterm

ID: 115200328

First name Last name Jimmy Chen

56

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays. ✓
- Store Data as 1D Array (10 points) ✗  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

#### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

11010011

115200328

Jimmy Chen

```
#include <stdio.h>
```

```
int main() {
```

```
FILE *in, *out;
```

```
float arr1[32], arr2[32], temp;
```

```
int i=0, j=0, z=0;
```

```
if ((in = fopen("input1.txt", "r")) != NULL) { //read and store input1.txt  
    while ((fscanf(in, "%f", arr1[i])) != EOF) {  
        i++;  
    }  
}
```

```
else {  
    printf("Invalid input file");  
}
```

```
fclose(in);
```

```
if ((in = fopen("input2.txt", "r")) != NULL) { //read and store input2.txt  
    while ((fscanf(in, "%f", arr2[j])) != EOF) {  
        j++;  
    }  
}
```

```
else {  
    printf("invalid input file");  
}
```

```
for (int y=0; y<j; y++) {  
    while (1) {  
        if (arr2[y] != arr1[z]) {  
            z++;  
        }  
        else
```

// store arr2 into arr1 without dupe.

merge! duplicates? - 10

Q1: -13+3

```

for (int x=0; x<i; x++) {
    for (int y=0; y<i-1-x; y++) {
        if (arr[y] > arr[y+1]) {
            temp = arr[y];
            arr[y] = arr[y+1];
            arr[y+1] = temp;
        }
    }
}

```

// bubble sort

```

3
out = fopen("outout.txt", "w");
printf(out, "Third ID array (Result) is:"); // print third array
for (int y=0; y<i; y++) {
    fprintf(out, "%f, ", arr[y]);
}
temp = 0;
for (int y=0; y<i; y++)
    temp = temp + arr[y];
printf(out, "\n sum is: %f", temp);
printf(out, "\n Average is: %f", temp/i); // print average
fclose(out);
return 0;
}

```

// print sum

// print average

$$\ln(x) = \sum_{k=1}^{\infty} \frac{(x-1)^k}{k}$$

$$\frac{(x-1)^{k+1}}{k+1} \cdot \frac{k}{(x-1)^k} = \frac{(x-1)k}{k+1}$$

115200324  
Jimmy Chan

```
#include <stdio.h>
```

```
int main() {
```

```
    int k = 1, n;
```

```
    float x, sum, term;
```

```
    sum = 0;
```

```
    printf("enter n term: "); // Ask for n and x
```

```
    scanf("%d", &n);
```

```
    getchar();
```

```
    printf("Enter value of x: ");
```

```
    scanf("%f", &x);
```

```
    term = (x-1) * k / (k+1);
```

```
    k++;
```

```
    for (k; k < n; k++) {
```

```
        term = term * (x-1) * k / (k+1);
```

```
        printf("%dth term of ln(%f) is: %f", k, x, term); // calculate each term
```

```
        sum = sum + term; // T_{k+1} = T_k * ratio, ratio is (x-1)k/(k+1)
```

```
    }
```

```
    printf("sum of ln(%f) on %dth term is: %f", x, n, sum);
```

```
    return 0;
```

```
}
```

Q2: -16

P2: -15

115200324 Jimmy Chen

```
#include <stdio.h>
```

```
int main() {  
    unsigned int inp, mask1, mask2;  
    int checksum=0, mask3, mask4;  
    printf("Enter a 8 bit binary number in hex: "); // Ask for 4 bit binary input  
    scanf("%x", &inp);
```

```
    mask1 = ((inp & 0xF8) >> 3); // isolate [7-3]
```

```
    mask2 = (inp & 3) // isolate [2-0]
```

```
    for (int check=0; check<4; check++) // do checksum
```

```
        mask3 = 1 << check;
```

```
        mask4 = mask1
```

```
        mask4 = mask4 & mask3
```

```
        if (mask3 == mask4) {
```

```
            checksum += 1;
```

```
        }
```

```
    }
```

```
    printf("Checksum is: %d", checksum); // print checksum
```

```
    return 0;
```

```
}
```

+10

115200328

Jimmy Chen

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

int main() {
    FILE *in, *out;
    string text[32], name[32];
    int i=0, j=0;
    in = fopen("input-sentence.txt", "r"); // store and read input sentence
    while ((fscanf(in, "%s", text[i])) != EOF) {
        i++;
    }
    fclose(in);

    in = fopen("names.txt", "r"); // store and read name file
    while ((fscanf(in, "%s", name[i])) != EOF) {
        i++;
    }
    fclose(in);
    for (int y=0; y<i; y++) {
```





## ESE 124 Fall 2023 Midterm

ID: 114859871

First name Last name Aameem Khan

62.5

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)**  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)**  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)**  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)**  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots - \frac{(x-1)^k}{k}$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).**

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

---

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

---

## Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

Q1

ID: 114859871

NAME: Ameer Khan

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#define MAX 30
```

Q1:-10

Int main ()

```
{
    File *f = fopen("input1.txt", "r"); // open first file
    File *f1 = fopen("input2.txt", "w"); // open second file
    File *f2 = fopen("output.txt", "x"); // open output file

    int i=0;
    float arr1[MAX];
    float arr2[MAX];
    fscanf(f, "%f", &arr1[i]);
    fscanf(f1, "%f", &arr2[i]);

    float arr3[MAX];

    while ((fscanf(f, f1, "%f %f", &arr3[i])) != EOF) // scan both files, take the
    { // values, and place them in arr3[i]
        if (arr3[i] == arr3[i+1]) // if any of the numbers are
        { // equal
            arr3[i] = ((arr3[i] + arr3[i+1]) / arr3[i]); // add them together
            i++; printf("arr[%d]", arr[i]); // and divide to only
        } // leave one of them.
        // Print the remainder

    }

    int length = i;
    for (int j=0; j < length; j++)
    {
        for (int b=0; b < (length-1); b++) // Sorting after setting
        { // length
            if (arr3[b] > arr3[b+1]) // ascending order
            {
                int tem = arr3[b];
                arr3[b] = arr3[b+1];
                arr3[b+1] = tem;
            }
        }
    }

    // bubble sorting
    // continue on back
```

} }

for (j=0 ; j < length; j++)

{ printf ("arr3[%d]" = %d, b, arr3[b]); }

// take the  
sorted  
array  
and print  
it.

-5

Q2 Ameen Khan  
ID: 114859871

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

```
int main ()
```

```
{
    int i=0;
    float x, T, sum;
    int n;
    printf("enter value for x ");
    scanf("%f", &x);
    getch();
    printf("enter # of terms n");
    scanf("%d", &n);
```

// get values for x and n

do

```
{
    T = (-1 * pow((x-1), i)) / i;
    printf("%f = %d term", T, i);
```

// set up loop to show each term for each iteration

```
    sum = T + sum;
```

// increment while showing each term ~~then~~ and sum up every term

```
    i++; }
```

```
while (i <= n)
```

```
printf("Sum = %f", sum);
```

// show the sum of all the terms.

```
return 0; }
```



Q3 Aneem Khan  
114859871

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 32
```

```
int main ( )
```

```
{
    File *f = fopen("input_sentence.txt", "r");
    File *fw = fopen("names.txt", "w");
    File *fo = fopen("output_sentence.txt", "z");
}
```

Q3=12.5

// open up files

7

-7.5

```
int i=0;
char sent[MAX];
char out[MAX];
char name[MAX];
// String from each file
```

```
fscanf(f, "%c", sent[MAX]); // store the input files into arrays/strings
fscanf(fw, "%c", name[MAX]);
```

```
while((fscanf(f, "%c", sent[MAX]) != EOF))
{
    if (strcmp(name[i], sent[i]) == 0)
    {
        sent[i] = sent[i] + 32;
        i++;
    }
}
```

// somehow trying to make it so that  
if segments of characters like  
up in both files, capitalize the  
first letter of the segment

-5

```
sent[MAX] = *fo; // transfer new sentence string  
fclose(*f); // to output file.
```

```
fclose(*fw); // closing all files
fclose(*fo);
```

```
Return 0; }
```





## ESE 124 Fall 2023 Midterm

ID: 115120484

First name Last name

Stanley Rohro

91

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:

Read words from an input file named "input\_sentence.txt".

Read names from an input file named "names.txt".

Create a new file named "output\_sentence.txt".

Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

**Q4: Bonus 10 points**

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011,  $1+1+0+1+0=3$ ).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

Starley Chro  
115120489

Q, :

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
// swaps two elements in array
```

```
void swap(float *a, float *b)
```

Q1:-4

```
{
```

```
    *a = *a + *b;
```

```
    *b = *a - *b; // (*a + *b) - *b == *a or *b = *a
```

```
    *a = *a - *b; // (*a) - (*a + *b) == -*b or *a = -*b
```

```
}
```

```
int main()
```

```
{
```

```
    FILE* f1; two files -2
```

```
    FILE* f2;
```

```
    // error-handle for open file
```

```
    if ((f1 = fopen("input1.txt", "r")) == NULL || (f2 = fopen("input2.txt", "r")) == NULL)
```

```
    {
```

```
        puts("File error");
```

```
        return 1;
```

```
    }
```

```
    FILE * out;
```

```
    if ((out = fopen("out.txt", "w")) == NULL)
```

```
    {
```

```
        puts("Write error.");
```

```
        return 1;
```

```
    }
```

```

// while loop to store f1 in a
float sum = 0; // for average, and sum
float a[60];

int aLen = 0;
while (fscanf(f1, "%f", &a[aLen]) != EOF)
{
    for (int i = 0; i < aLen; i++)
    {
        if (a[aLen] == a[i]) // exists? Don't iterate iter for. Overwrite.
            aLen--; a[i] = NULL; break; // run outta space
        aLen++;
    }
    fclose(f1);
}

// while loop to store f2 in b
int bLen = 0;
float b[30];
while (fscanf(f2, "%f", &b[bLen]) != EOF)
{
    for (int i = 0; i < bLen; i++)
    {
        if (b[bLen] == b[i]) // exists? Don't increment bLen. subtraction.
            bLen--; b[i] = NULL; break; // run outta space
        bLen++;
    }
}

float merged[60];
for (int i = 0; i < aLen; i++)
{
    merged[i] = a[i];
}

for (int i = 0; i < bLen; i++)
{
    merged[i + aLen] = b[i]; // offset for merge, and odd
}

int sorted = 0;
while (!sorted)
{
    sorted = 1; // V this is sum of lengths a, and b. Add to get merge
    for (int i = 0; i < aLen + bLen - 1; i++)
    {
        if (merged[i] > merged[i+1])
        {
            swap(&merged[i], &merged[i+1]); // swap bubble
            sorted = 0;
        }
    }
}

```

// print array

Stanley 6020  
115120484

for (int i = 0; i < a.len + b.len; i++)

{  
 sum += merged[i];  
 printf("%d ", merged[i]);  
}

}

printf("sum : \n Average : ", sum, sum/  
↳ (a.len + b.len));

return 0;

-2 write in files

}

Q 4.

~~#include <stdio.h>~~

~~int main~~



Stanley Gohro

115120489

Q2:

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

```
int main() {
    float tempsum; // part 2
    float sum;
```

```
    float x;
    // user input
    puts("x: ");
```

```
    scanf("%f", &x);
```

```
    puts("n: ");
```

```
    scanf("%d", &n);
```

```
    printf("1. (%f) = (%f-1)",
```

```
        x, x);
    int min_index = 0; // part 2
    int min = x-1;
```

```
    float term = x-1;
```

```
    sum = x-1; // start
```

```
    for (int i = 1; i < n; i++)
```

```
    {
        term = -1 * i / ((x-1) * (i+1));
        // add or subtract
```

```
        if (i % 2 == 0)
```

```
        {
```

```
            printf("- ");
```

```
        } else {
```

```
            printf(" ");
```

```
            printf("(%f-1)^%d", x, x, x);
```

```
            sum += term;
```

```
            tempsum += term;
```

```
        if (tempsum < min && i >= 3) // find smallest
```

```
        {
```

```
            min = tempsum;
```

```
            min_index = i;
```

```
        }
    }
    return 0; }
```

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1} \cdot (x-1)^n}{n}$$

$$\frac{(-1)^{n+1} \cdot (x-1)^{n+1}}{n+1}$$

$$\frac{(-1)^{n+1} \cdot (x-1)^{n+2}}{n}$$

$$\frac{(x-1)^n \cdot (x-1)}{(x-1)^n \cdot (x-1) \cdot (x-1)}$$

$$-\frac{1 \cdot n}{(x-1)(n+1)}$$

// for part 2,  
it should be  
infinity

-5

Q 3.

115 120484

Stanley QW D

```

#include <stdio.h>
#define MAX_STR 100;
int main() {
    FILE *names;
    FILE *in;
    FILE *out;
    // handle err
    if ((in = fopen("input-sentence.txt", "r")) == NULL ||
        (names = fopen("names", "r")) == NULL ||
        (out = fopen("output-sentence.txt", "w")) == NULL)
    { puts("Error file");
    }
    int i = 0;
    char name[MAX_STR];
    char word[MAX_STR];
    while (fscanf(in, "%s", &name) != EOF) // stop name in line
    {
        word[0] = 32; // capitalize letter
        while ((word = fgets(in)) != EOF) // read string
            if ((strcmp(word, name)) != 0) // not equal
            {
                word[0] += 32; // put it back bcz !=
            }
        fprintf(out, "%s", word);
    }
    fclose(names);
    fclose(in);
    fclose(out);
    return 0;
}

```



Starkey

Q4.0

15120 989

Q4.

```
#include <stdio.h>

int main()
{
    char par = 42; // made up. It said so!
    char checksum = 0;
char data;
    char mask = 0xf8;
    char data = par & mask; // + 10
    for (int i = 0; i <= 6; i++) // loop 7, 3
    {
        if (data & 0x1) // find 0 or 1 LSB
        {
            checksum += 1;
        }
        data = data >> 1;
    }
    printf("checksum %d, data %d", checksum, data);
    return 0;
}
```



## ESE 124 Fall 2023 Midterm

ID:

First name Last name

Fazlokh Fayzullaev

15

59.5

$$arr[n] = arr[n+1]$$
$$arr[n+1] = arr[n+2]$$

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is {2.0 3.0 4.0 5.0 7.0 8.0} = arr[0:7]

Sum is: 29.0

Average is: 4.83

Sum of 'arr[i]'

**Steps:**

- a) Read Data (10 points)

Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.

- b) Store Data as 1D Array (10 points)

Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.

- c) Using Bubble Sort (10 points)

Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.

- d) Write the Solution to Output File (5 points)

Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."

Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

$$(x-1)^i$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- a) Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

for  $C = 0$  to  $3$   $C++$   
 $C++ <$

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

#### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
 Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

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

Q1: -11.5

```
int main() {
```

```
FILE * file1 = fopen("input1.txt", "r");
FILE * file2 = fopen("input2.txt", "r");
```

} // opening

```
if (file1 == NULL || file2 == NULL);
printf("cannot work!");
```

} // exception handler

```
return 0; }
```

```
double arr1[30];
double arr2[30];
double merged[60];
```

} // limited to 30 for array 1  
} // and times 2 for merge 1

```
int n, m;
```

```
scanf("%d", &n); // size of arr1
scanf("%d", &m); // size of arr2
```

float -0.5

merging into 1 array

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

```
scanf("%f", &arr1[i]);
arr1[i] = merge[i]; }
```

} // scans file 1 and sets for merge equal.

```
for (int j = 0; j < m; j++) {
```

```
arr2[j] = merge[i];
i++; }
```

} // scans file 2 sets equal to the continuation of merge[i]

duplicates removal! -5

```
} fclose(file1);
fclose(file2);
```

// closing files

```

void bubbleSort
int temp;
for (int i=0 ; i < n-1 ; i++) {
    for (int j=0 ; j < n-1-i ; j++) {
        if (merge[j] < merge[j+1]) {
            temp = merge[j]
            merge[j] = merge[j+1]
            merge[j+1] = temp
        }
    }
}

```

✓

// Bubble  
Sort  
Algorithm

```

if (merge[i] == merge[i+1]) {
    merge[i] = merge[i+1]
}

```

// eliminates  
duplicates.

```

for (int f=0 ; f < i ; f++) // reads until
                             the size of
                             merge[i]. i

```

```

float sum = merge[i+1] merge[i+1];

```

```

float sum = merge sum + merge[i];

```

```

printf ("sum of the array is \"%s\"", sum);

```

```

float average = sum / i; // sum divided by
                          total storage

```

```

printf ("average is \"%s\"", average);

```

```

return 0;

```

```

}
}

```

-2. write to output file.

Q 2.

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

```
int main () {
    int n;
    printf
    int i = 1;
    double term = (x-1);
    double result = 1.0;
    double x;
```

} declaration  
of variables

```
printf ("input your x term in "); // input for term
scanf ("%d", &n);
```

```
for (int i = 0; i < n; i++) {
    term = -term * (x-1) / i; // i increases and power (x-1)^n increases
    result += term; // next term calculation for series.
    i++; // moves to the next term by loop
}
```

term = ~~abs~~ int arr[n]; - part II

```
for (int i = 0; i < n; i++) {
    if (abs(arr[n+1] + arr[n+2] + arr[n+3]) < term) {
```

```
term = abs(arr[n]) // increasing
    n++;
    printf ("%f", n);
    return 0;
```





Q 3)

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
```

char word[100];  
char name[100];  
int words = 0;

```
FILE * sentence = fopen("input_sentence.txt", "r");
// opening
// exception handling
if (sentence == NULL) {
    printf("error\n");
}
FILE * Names = fopen("names.txt", "r"); // open file
if (file == NULL) {
    if (Names == NULL) {
        printf("\n error");
    }
}
// exception handling
```

```
while (fscanf(sentence, "%s", word) != EOF) {
    for (fscanf(sentence, "%s", &word) != EOF);
    word[0] = '\0';
    if (strlen(word) > 0) {
        L++;
    }
    while (fscanf(Names, "%s", &name) != EOF) {
        for (int i = 0; name[i] != '\0'; i++) {
            name[i] = toUpper(name[i]); // makes upper for letter
        }
    }
}
```

```
FILE * Out = fopen("Output_sentence.txt", "w");
if (Out == NULL) {
    printf("error\n");
}
```

while

```
fprintf(Out, "%s", word, name);
```

-4



## ESE 124 Fall 2023 Midterm

ID: 115086584

First name Last name Qi Rui Jiang

68

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0      2 2 3 4  
second 1D array is 8.0 5.0 7.0 2.0      2 5 7 8

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)**  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)**  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)**  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)**  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:

Read words from an input file named "input\_sentence.txt".

Read names from an input file named "names.txt".

Create a new file named "output\_sentence.txt".

Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

#### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

```

#include <string.h>
Q1: #include <stdio.h>
#include <stdlib.h>
int main() { #define len 30
FILE * inP_f = fopen("first.txt", "r");

```

QiruiJiang

115086588

// read data

Q1:-6

```

FILE * inP_f1 = fopen("second.txt", "r");
FILE * out_f = fopen("third.txt", "w");

// variables? arrays? -2
if (inP_f == NULL || out_f == NULL || inP_f1 == NULL) { // check for errors
    printf("Error"); }

```

```

float n[len], m[len], temp, temp2;
while ((fscanf(inP_f, "%d", &n)) != EOF) { // scan 1st input

```

```

    printf("%d", n); n++; }

```

-2 third array?

```

while ((fscanf(inP_f1, "%d", &m)) != EOF) { // scan 2nd input
    printf("%d", m); m++; }

```

```

for (i=0; i<len; i++) { // use bubble sort to sort 1D array

```

```

    for (int j=0; j<len-i-1; j++) {

```

```

        if (n[j] > n[j+1]) {

```

```

            temp = n[j]; // store in temp

```

```

            n[j] = n[j+1]; // swap values from small to large

```

```

            n[j+1] = temp; } } }

```

continue

```
for(m=0, m<len<m++) { // bubble sort 2nd array
```

```
for(int k=0; k<len-k-i; k++){
```

```
if(m[k]>m[k+1]){
```

```
    tmp2 = m[k]; // swap value from small to large
```

```
    m[k] = m[k+1];
```

```
    m[k+1] = tmp2; }
```

```
} }
```

```
if(n[i]!=m[k]) {
```

// Compare values of each array

```
    m[k]=n[i+len];
```

// If not the same, add into 1<sup>st</sup> array

```
    i++;
```

// Increment i, so it doesn't override n[i].

```
}
```

```
int sum=0; float avg;
```

```
for(i=0, i<len; i++){
```

```
    sum = sum + n[i]; // add the arrays together
```

```
fprintf(out_p "%f", avg/i);
```

```
fprintf(out_p "%d", sum); // sum
```

```
fprintf(out_p, "%d", n[i]); // order form "
```

```
} // i++?
```

```
fclose(inp-f);
```

```
fclose(inp-fi);
```

```
fclose(out-p);
```

```
}
```

-2

2.

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main () {
```

```
int x, k=1, n;
```

```
float eps;
```

```
printf("Enter a value for x: "); // x value in Taylor series,
scanf("%d", &x);
```

```
printf("Enter level of precision"); // level of precision
scanf("%f", &eps);
```

```
do { float sum = x-1;
    n = (x-1) * -1 / (k+1); -2 // gives the next term of the function.
    k++;
```

```
sum = sum + n; // add the term with ratio of next term.
```

```
printf("%dth term is: %f", k, sum); // Print the 1st to kth term
```

```
while(fabs(eps) > 0.000001);
```

```
printf("The solution is %f", sum);
```

```
printf("The %d, %d, %d, th term have the smallest sum"; k, k-1, k-2);
```

$$\frac{t_{k+1}}{t_k} = -\frac{(x-1)^{\frac{1}{2}}}{2} \times \frac{1}{x(x+1)} =$$

$$\frac{-(x-1)^{\frac{1}{2}}}{2} \text{ next term}$$

$$\text{start at } \frac{(x-1)^k (-1)^k}{k}$$

$$k=1 \text{ first}$$





```
#define Len 30
#include <stdio.h>
#include <stdlib.h>
```

```
FILE *inp_f = fopen("Input.txt", "r"); // read from wire files
```

```
FILE *inp_f1 = fopen("name.txt", "r");
```

```
FILE *out_f = fopen("output.txt", "w");
```

```
int main() {
    char In[STR], Name[STR];
    int i, j;
```

```
while((fscanf(inp_f, "%s", &In)) != EOF) { // store Input data
    printf("%s", In[i]);
    i++; }
}
```

```
while((fscanf(inp_f1, "%s", &Name)) != EOF) { // store Name data,
    j++;
```

```
printf("%s", Name[j]); }
}
```

-10

```
for(i=0; i < Len; i++) { // check each word to see if it is in the name file.
```

```
for(j=0; j < Len; j++) {
```

```
if(In[i] == Name[j]) { // change 1st letter if seen in name file.
```

```
In[i] = 'A', 'Z'; }
}
```

```
}
```

```
fprintf(out_f, "%s", In[i]); // outputs to output.txt
```

```
fclose(inp_f);
```

// Close the file

```
fclose(inp_f1); fclose(out_f); }
```



A=10  
B=11  
C=12

84 1 13  
1101 = 0  
0011 = 3

4.

```
int main() {  
    int Var, shift, i; // 8 bit Variable  
    int mask = 0x1f, count = 0;  
    printf("Enter any 8 bit variable:");  
    scanf("%x", &Var);
```

```
    shift = Var >> 3; // shift 3 time to right, clear out last 3 bits
```

```
    // And the Var with a mask
```

```
    Var & mask;
```

00010000

```
    if
```

+1



Keller Chw 11510032

#1 include <stdio.h> #include <string.h> #define STR\_LEN 32

Q1: main()

FILE \*f1, \*f2, \*f3;

f1 = fopen("input-file1.txt", "r"); // opens first input file for reading

f2 = fopen("input-file2.txt", "r"); // opens second input file for reading

f3 = fopen("output-file.txt", "w"); // opens output file for writing

float arr1[STR\_LEN], arr2[STR\_LEN], tmp;

int count1 = 0, count2 = 0, x, y, i, j, k;

float sum = 0, average;

if (f1 == NULL || f2 == NULL || f3 == NULL) // check if file  
// file can be opened

{ printf("File cannot be opened!\n");

exit(1);

strcpy(f1, f2) // copy file -10 +5

if (

// If f1 has two (f1 & f2) same numbers per row of  
one of them (don't know how to code this)

while (fscanf(f1, "%f", &arr1[count1]) != EOF)

{ count1++; // increment count of number read -6 +4

}

for (x = 0; x < count1; x++)

← bubble sort

{ for (y = 0; y < count1 - 1 - x; y++)

{ if (arr1[y] > arr1[y+1])

{ tmp = arr1[y];

45 + 10  
+ 23  
- 5

73

Q1: -18

```
arr[x] = arr[x+1];
```

```
arr[x+1] = temp;
```

```
};
```

```
}
```

```
}
```

```
for (x=0; x < Count; x++)
```

```
{ printf("The third Element (result) is %.2f\n", arr[x]);
```

```
for (i=0; i < Count; i++)
```

```
{ for (j=i; j < Count; j++)
```

Summing of terms

```
Sum = 0;
```

```
for (k=1; k <= j; k++)
```

```
{ Sum = Sum + arr[k];
```

```
};
```

```
};
```

```
};
```

```
printf("Sum is %f", Sum);
```

```
Sum / Count = average;
```

Averaging of terms

or print

```
printf("Average is %f", average);
```

```
return 0;
```

-2

```
fclose(f3);
```

```
fclose(f2); // closes files
```

```
fclose(f1);
```

Keller Chu 115106332

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main()
```

```
{ float n, x, ratio, term, sum, smallest, second, third;
```

```
printf("Enter value of x:"); // prompt for value of x
```

```
scanf("%f", &x);
```

```
printf("Enter number of iterations for loop:"); // prompt for value of n
```

```
scanf("%f", &n);
```

```
term = -1;
```

```
sum = 0;
```

init y=1;  
// initialize  
counter

```
do
```

```
{ ratio =  $(x-1)^*$  / (n / (n+1)); // ratio for next term
```

-5

```
term = term * ratio; // calculate term
```

```
sum = sum + term; // add term with previous sum
```

```
y++; // increment of the counter
```

```
} while (y <= n); // continues until n iterations
```

```
smallest = (pow(-x-1, n) / n); // always the smallest
```

```
second = (pow(-x-1, n-1) / (n-1)); // always the second
```

```
third = (pow(-x-1, n-2) / (n-2)); // always the third
```

-14+4

```
printf("ln(%f) with %f iterations = %f", x, n, sum);
```

```
printf("The three consecutive terms that have the smallest sum is:
```

```
%f, %f, and %f", smallest, second, third);
```

```
return 0;
```

```
}
```





Kellan Chui 115196332

Q3: #include <stdio.h> #include <string.h> #include <ctype.h>  
int main()

#define LEN 64

initialise?

FILE \*f1, \*f2, \*f3;

-3

f1 = fopen("input-sentence.txt", "r"); // reads

f2 = fopen("names.txt", "r"); // reads

f3 = fopen("output-sentence.txt", "w"); // writes

if (f1 == NULL || f2 == NULL || f3 == NULL) // Checks for file

printf("File not found!\n");

exit(0);

// closes all files and returns

for

word[LEN], names[LEN];

fscanf(f1, "%s", word) != EOF; // array to hold

fscanf(f2, "%s", names) != EOF; //

if (strcmp(word, names) != 0) // Compare the two

upper(j); // uppercase

upper(j); // uppercase

// Don't know how to properly  
find / replace and change characters.

strcpy(f3, f1); // copy f1 with new capitalization to f3/output file

return 0;

-5

-5 (r.w)



## ESE 124 Fall 2023 Midterm

ID: 115180370

First name Last name Jefferson Panora

36 + 3 + 8  
47

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0  
second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0  
Sum is: 29.0  
Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates.  
Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array.  
Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
  - c) Sum to the terms by using a loop. (5 points)
- Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

**Q4: Bonus 10 points**

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

Handwritten calculations for Q4:

11010011  
 7 6 5 4 3 2 1 0  
 1101 0001  
 3 2  
 10100011  
 0000 0011

1) #include <stdio.h>  
#include <stdlib.h>  
#define MAX 100

int main()  
{ int length = 6  
float sum;

FILE \*f; // introducing the file

int value [MAX], a; // declaring functions

int i = 0, b, is\_swap; // Declaring functions

if ((f = open("input.txt", "r")) == NULL) // setting if statement to open input.txt in error  
{ print("Error, cannot open file"); // NULL condition.  
}

FILE \*f0 = ((f = open("input 2.txt", "r"))); // introducing second input file.

FILE \*f1 = ((f = open("input 3.txt", "r")));

FILE \*f1 = FILE \*f0 + FILE \*f // Merges the files to sort.

for (int i = 0; i < length; i++)

{ for (int j = 0; j < (length - 1); j++)

{ if (a[j] > a[j+1])

{ int temp = a[j]

a[j] = a[j+1]

a[j+1] = temp;

if (a[j] == a[j+1])

{ printf("%d ", a[j]);

// This means that after sorting if the variables next to each other are equal, a space prints instead.

fclose(input 3.txt);  
fclose(input.txt);

~~10~~ ~~5~~ ~~5~~  
-10 -5 -5

Q2) #include <stdio.h>  
#include <math.h>

int main()

{

float x, a;

float sum = 0; // Will start counting from 0.

int n;

printf("Enter a value:\n"); // Entering my x value

scanf("%f", &x);

getchar() // to avoid scanf issues

printf("Enter value for n:\n");

scanf("%d", &n);

getchar() // avoid scanf issues

x = (x-1); // This computes x-1 so it can then be raised to the n power

for (int i = 0; i < n; i++) // n counts the term so if i < n then the system will only run for n terms v.

{ sum += pow(x, n) / n; // calculates x to the power of n over n for n times. ✓

}

printf("Taylor series: %f and Term: %d", sum, n);

// This will print the value at the n term.

return 0;

}

I'll take this as

P2 Since ↓ is better solution for P1.

Q2 Part 2)

#include <stdio.h>

int main()

float x, t, sum; // declaring functions

int n, k;

printf("x = ");

scanf("%f", &x);

getchar() // to avoid scanf errors next page.

```
printf("n = \n"); // getting my functions
scanf("%d", &n);
getchar();
```

```
+
sum = t;
for (k=0, k < n, k++) // setting my condition
{ t = t * (-1) * (x-1.0) * (k+1) / (k+2); }
sum = sum + t;
}
printf("ln %f = %f", sum); }
```

~~115180370~~

Q3) #include <stdio.h>

#include <stdlib.h>

int main() char c;

{ FILE \*f;

f = fopen("input\_sentence.txt", "r"); // Opening files to write sentences

f = fopen("names.txt", "w");

if (c == ' ' || c >= 'a' || c <= 'z' || c >= 'A' || c <= 'Z') // identify

{ c -= 32; // This is to identify uppercase and lowercase letter, Ascii value difference between upper and lower is 32!

fputc(c, f);

c = fgetc(inp);

~~0/35~~

~~-30+5~~

printf("Please Have Mercy\n"); ?

return 0;

## Q4 Bonus

```
char b;
char hexa;
char a;
int z;
int sum;
{ printf("Enter a number in hexa:\n");
  scanf("%x", &hexa);
```

// I need to enter in hexa value.

```
a = hexa & 0xf8;
```

// This makes bits 7 to 3 stay the same while changing the rest to 0, then we can count for 1s.

// 11010 011  
x 1111 000  
↑ f 8

```
for (int i=0; i<7; i++)
{ count 1st
  sum = count 1;
}
```

// This makes i increment when it sees one, and since there is only 7 times it is set to <7.

```
b = a & 0x03;
```

// This returns a to the [2,0] format to check.

```
printf("%x=%d", b, z);
```

// This converts the hexadecimal to decimal and will print 3.

```
printf("%d", sum);
```

// This will print the sum = 3.

```
return 0;
```

```
}
```

+10-2



## ESE 124 Fall 2023 Midterm

ID: 113245284

63

First name Last name

Pietro (Pete) (Munoz) Foreman

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array**. The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.  $\ln x = \sum_{i=1}^n \frac{(x-1)^i \cdot (-1)^{i+1}}{i}$

- Define the variables and initial them (5 points).

ratio =

$$\frac{(x-1)^n \cdot (-1)^{n+1}}{n} \cdot \frac{n}{(x-1)^n \cdot (-1)^{n+1}}$$

$$t_{k+1} = \ln(x+1) = \sum \frac{(x)^n \cdot (-1)^{n+1}}{n}$$

$$\text{ratio} = \frac{1}{(-1)^n}$$

- b) \* Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

#### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

```
1a) #include <stdio.h>
# define SIZE 30
```

```
int main()
```

```
{
```

```
FILE *f1, *f2, *f3;
```

```
f1 = fopen("input1.txt", "r"); // opening files; reading inputs, writing to output
```

```
f2 = fopen("input2.txt", "r");
```

```
f3 = fopen("output.txt", "w"); // open & write to output file
```

```
int i1 = 0, i2 = 0, i = 0;
```

```
float arr1[SIZE], arr2[SIZE], arr[60];
```

```
float temp;
```

```
while (fscanf(f1, "%f", &arr1[i]) != EOF) // scan f1 to arr1
```

```
{
```

```
    i1++;
```

```
}
```

```
while (fscanf(f2, "%f", &arr2[i2]) != EOF) // scan f2 to arr2
```

```
{
```

```
    i2++;
```

```
}
```

```
for (int c = 0; c < SIZE; c++) // copying arr1 & arr2 to arr
```

```
{
```

```
    arr[c] = arr1[c];
```

```
    for (int j = 0; j < SIZE; j++)
```

```
    {
```

```
        arr[c+59] = arr2[j];
```

```
    }
```

```
}
```

total 60?

-2

```
for (int c = 0; c < 60; c++)
```

-5 duplicates?

```
{
```

```
    for (int j = 0; j < 60 - c; j++)
```

```
    {
```

```
        if (arr[j] < arr[j+1])
```

→ cont

Q1:-7

cont  
→

```

    if (arr[j] < arr[j+1])
    {
        temp = arr[j]; // get element
        arr[j] = arr[j+1]; // skip
        arr[j+1] = temp; // swap
    }
}

float sum = 0;
for (int x = 0; x < 60; x++)
{
    fprintf(f3, "%f", arr[x]); // print array to output
    sum += arr[x];
}

fprintf(f3, "\n Sum is : %f \n Average is : %f", sum, sum/60); // print to output
fclose(f1); // close files
fclose(f2);
fclose(f3);
}

```

```

2 #include <stdio.h>
  #include <math.h>

int main()
{
    int n;
    float x, sum = 0;

    printf("Enter H of terms:"); // scan in values
    scanf("%d", &n);
    printf("Enter x value: ");
    scanf("%f", &x);

    if (n % 2 == 0) // determine next x
        next = x;
    else
        next =  $(-1) \cdot x$ ;

    for (int i = 0; i <= n; i++) // summing terms
    {
        sum +=  $x / \text{pow}(-1, i)$ ;
    }
    return 0;
}

```

$-5 - 5 - 15$

```
#include <stdio.h>
#include <string.h>
#define SIZE
```

```
int main()
```

```
{
```

```
FILE *f1, *f2, *f3;
```

```
f1 = fopen(f1, "input-sentence.txt", "r"); // open files
```

```
f2 = fopen(f2, "input-names.txt", "r");
```

```
f3 = fopen(f3, "output-sentence.txt", "w");
```

```
if (f1 == NULL || f2 == NULL || f3 == NULL) // handle error
```

```
{
    printf("Failed to open file\n");
```

```
    exit(1);
```

```
}
```

```
char names[SIZE], sentence[SIZE]; // create strings
```

```
int len_name = 0;
```

```
while (fscanf(f1, "%s", &names[len_name]) != EOF) // read names, get len
```

```
{
```

```
    len_name++;
```

```
}
```

```
len_name int len_sen = 0;
```

```
while (fscanf(f2, "%s", &sentence[len_sen]) != EOF) // read words, get len
```

```
{
```

```
    len_sen++;
```

```
}
```

```
for (int i = 0, i < len_name; i++) // make names capital
```

```
{
```

```
    strcpy(&len_names[i], toupper(&len_names[i])); == 0
```

```
}
```

-5

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

```
{
```

```
    if (strcmp (names[i], sentence[i])
```

```
        fprintf (f3, "%s", names[i]); // if name, print capital from name every
```

```
    else
```

```
        fprintf (f3, "%s", sentence[i]);
```

```
}
```

```
fclose (f1);
```

```
fclose (f2);
```

```
fclose (f3);
```

```
return 0;
```

```
}
```





## ESE 124 Fall 2023 Midterm

ID: 114908803

39 +10+3

First name Last name

Rosha Ramnane

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) \* Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:

Read words from an input file named "input\_sentence.txt".

Read names from an input file named "names.txt".

Create a new file named "output\_sentence.txt".

Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

*Read as str*

*char + 32*

*Read words as array*

*+ print str with arr[1] and arr[2]*

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**rank are taking an exam.*

#### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011,  $1+1+0+1+0=3$ ).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

// Roshen  
 // Ramnante  
 // 114908803  
 // Q1

```
#include <stdio.h>
int n, n1;
int sum = 0;
int main() {
  FILE *f1 = fopen("first-input-array.txt", "r"); // opens 1st array, specifies array
  FILE *f2 = fopen("second-input-array.txt", "r"); // opens 2nd array, specifies array
  FILE *f3 = fopen("Final-output-array.txt", "w"); // opens output array file
  n = (size of (first[])/size of (first[0])); // n = (size of (second[])/size of (second[0])); // ran out of room
  n1 = (size of (second[])/size of (second[0])); // n and n1 are terms
  for (int i=0; i < n; i++) { // present
    int first-array, second-array; // forgot to initialize
    fscanf(f1, "%d", &first-array);
    for (int i=0; i < n1; i++) {
      fscanf(f2, "%d", &second-array); // scans both files for arrays
      Third-array[i] = first-array + second-array; // Appends both arrays?
    }
    for (int j=0; j < (n+n1); j++) {
      if (Third-array[j] > Third-array[j+1]) {
        int temp = Third-array[j];
        Third-array[j] = Third-array[j+1];
        Third-array[j+1] = temp;
      }
    }
    // bubble sorts new array
    for (int k=0; k < (n+n1); k++) {
      if (Third-array[k] == Third-array[k+1]) {
        Third-array[k] = NULL; // deletes duplicates from our new appended
        // array by setting previous entry as NULL
      }
    }
    n3 = (size of (Third-array[])/size of (Third-array[0])); // scans for new amount of terms in output
    while (int k=0; k < n3; k++) {
      fprintf(f3, "%d\n"); // prints out to file
    }
    return 0;
  }
}
```

Q1: -16+2+5+3

-5

X

-1



```

// Roshen
// Ramenke
// 114908803
// Q2
#include <stdio.h>
#include <math.h>
float x;
int n;
float e = 0;
float sum = 0;
int main() {

    printf("Please input an x-value for ln(x): "); // Gets input for x
    scanf("%f", &x);
    getch();
    printf("\n Please input number of terms to run: "); // Gets input for n
    scanf("%d", &n);

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

        e = pow((x-1), i); // sets (x-1) to i power

        sum = sum + e/i; // adds back previous sum as it iterates
    }

    printf("\n Output of ln(%f): %f", x, sum); // computes summation of ln(x)
}

```

P2: -15



// Roshan

// Rammarine

// 11408803

// Q3

// The files need to be scanned in two data types. Input-sentence will become str

// Names needs to read as a str array to append the names.

// The names file should upper case the names first, by finding the words using

// fgetsc("\ ") to find a space between names. Input and Names.txt need strcmp

// to compare word placement, and then place the matching name in Input from names.txt.

// Sadly I am out of time.

:)

#include <string.h>

#include <stdio.h>

int main() {

FILE \*input = fopen("input-sentence.txt", "r");

FILE \*names = fopen("names.txt", "r");

fgetsc(names, "%c") {

// scans for spaces in names file

char = c;

if (c == " ") {

-30+3





## ESE 124 Fall 2023 Midterm

ID:

First name Last name

PURAN SADHU

115184048

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates.  
Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array.  
Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:

Read words from an input file named "input\_sentence.txt".

Read names from an input file named "names.txt".

Create a new file named "output\_sentence.txt".

Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

### Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

Q1:

PURAN SADHU

115184048

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 32
```

```
int main() {
```

```
FILE *input_file1, *input_file2, *output_file;
```

```
float arr1[SIZE], arr2[SIZE], arr3[SIZE], temp;
```

```
int i=0, j=0,
```

```
if ((fopen("input_file1.txt", "r")) != NULL) {
    printf("Cannot read input_file1");
    exit(1);
}
```

```
if ((fopen("input_file2.txt", "r")) == NULL) {
    printf("Cannot read input_file2");
    exit(1);
}
```

```
if ((fopen("output_file.txt", "w")) == NULL) {
    printf("Cannot write output_file");
    exit(1);
}
```

```
// THESE TWO WHILE LOOPS PUTS BOTH ARRAYS INTO ONE SO ITS EASY TO SORT THROUGH.
```

```
while ((fscanf(input_file1, "%f", &arr1[i])) != EOF) {
    arr3[i] = arr1[i];
    i++;
}
```

```
}
```

Q1: -0

good!

\*continued on next page\*

```
while ((fscanf(input_file2, "%f", &arr2[j])) != EOF) {
    arr3[i+j+1] = arr2[j];
    j++;
}
```

```
}
```

```
int x=0, y=0; num;
```

```
// BUBBLE SORT FOR THE THIRD ARRAY.
```

```
for (x=0; x<
```

```
if (i >= j) { num = i + j + 1;
    num = i;
```

```
} else {
    num = j;
```

```
}
```

```
for (x=0; x<num; x++) {
```

```
    for (y=0; y<num-1-x; y++) {
```

```
        if (arr3[y] > arr3[y+1]) {
```

```
            temp = arr3[y];
```

```
            arr3[y] = arr3[y+1];
```

```
            arr3[y+1] = temp;
```

```
        }
```

```
    }
```

```
}
```

```
// DELETING THE DUPLICATE VALUES FROM ARRAY.
```

```
int z=0; a=0;
```

```
for (z=0; z<num; z++) {
```

```
    if (arr3[z] == arr3[z+1]) {
```

```
num-1; → for (a=0; a<num; a++) {
```

```
    arr3[a] = arr3[a+1]
```

```
}
```

```
}
```

\* entered on next case \*

```
float sum, average;
```

```
sum = 0;
```

```
average = 0;
```

```
int b = 0,
```

```
for (b = 0; b < num; b++) {
```

```
    sum = sum + arr3[b]
```

```
}
```

```
average = sum / num;
```

```
fprintf("output-file.txt", "THE third 1D array (result) is %.f", arr3[0]);
```

```
fprintf("output-file.txt", "sum is: %.f", sum);
```

```
fprintf("output-file.txt", "average is: %.f", average);
```

```
fclose(input_file1);
```

```
fclose(input_file2);
```

```
fclose(input_file output_file);
```

```
return 0;
```

```
}
```

Q2:

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

$$\frac{(x-1)^n}{n}$$

```
int main() {
```

```
float sum=0, temp, sum3;
```

```
int x, n, i=0; // GETS INPUT FROM USER.
```

```
printf("ENTER A VALUE FOR X: ");
```

```
scanf("%d", &x);
```

```
printf("Enter a value for n: ");
```

```
scanf("%d", &n);
```

```
temp = -1;
```

```
sum = sum + temp;
```

```
n = 1;
```

```
// FINDS THE SUM FOR N TERMS.
```

```
do {
```

```
temp = pow((x-1), i);
```

```
temp = (pow((x-1), i)) / i;
```

```
sum = sum + temp;
```

```
i++;
```

```
} while (i <= n)
```

```
sum3 = sum;
```

```
i = 0;
```

```
// CHECKS FOR SMALLEST SUM OF THREE CONSECUTIVE TERMS
```

```
float sum2; sum2 = sum;
```

```
while (i <= n) {
```

```
temp = ((pow((x-1), i)) / i) + ((pow((x-1), i+1)) / (i+1)) + ((pow((x-1), i+2)) / (i+2));
```

```
sum = temp
```

```
if (sum2 < sum) {
```

```
i++;
```

```
} else if (sum < sum2) {
```

```
sum2 = sum
```

```
i++;
```

```
}
```

~~-10~~

P2: -15

\*continued on next page\*

~~Q2~~

```
printf("ln(%d) is equal to: %f", x, sum3);
return 0;
}
```

Q3:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char arr[256];
    FILE *input_sentence, *names, *output_sentence;
    //CHECKS IF FILES CAN BE READ.
    if ((fopen("input_sentence.txt", "r")) == NULL) {
        printf("Cannot read input file");
        exit(1);
    }
    if ((fopen("names.txt", "r")) == NULL) {
        printf("Cannot read input file");
        exit(1);
    }
    //CHECKS IF FILE CAN BE WRITE.
    if ((fopen("output_file.txt", "w")) == NULL) {
        printf("Cannot write output file");
        exit(1);
    }
    while ((fscanf(names, "%s", &arr[i])) != EOF) {
        IDK \_(ツ)_/_;
    }
}
```

-29





Q4:

```
#include <stdio.h>
```

```
int main() {
```

```
    int par = 11010011;
```

```
    int a, b;
```

```
    a = par >> 3;
```

```
    b = par << 5;
```

+/



## ESE 124 Fall 2023 Midterm

ID: 115069679

58 + 3

First name Last name

Tyler Hansen

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array**. The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

## Q4: Bonus 10 points

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

Tyler Hansen 115669679

Q1:

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

```
#define Max 30 // makes max length 30 numbers
```

```
int main ()
```

```
{
```

```
    File *input1, *input2, *output; // sets files
```

```
    fopen (input1, "input1.txt", "r");
```

```
    fopen (input2, "input2.txt", "r"); // opens all files
```

```
    fopen (output, "output.txt", "w");
```

```
    if (input1 == NULL || input2 == NULL || output == NULL)
```

```
        // if the files == NULL, we can't open them i.e. no file
```

```
        { printf ("cannot open file.");
```

```
          exit (1); }
```

```
    float arr1[Max], arr2[Max], arr3[Max], temp[Max] // sets arrays
```

```
    float sum=0, avg=0; // sets sum=0 initially and avg=0 initially
```

```
    int i=0, len=0, j=0; // for increments in for loops
```

```
    fscanf (input1, "%f", arr1[i]); // scans arrays
```

```
    fscanf (input2, "%f", arr2[i]);
```

```
    if (arr1[i] > arr2[i]) // stores into output
```

```
    { fprintf (output, "%f", arr1[i]);
```

```
      i++;
```

```
    }
```

Q1:-15

-2

-10

duplicates?

// bubble sort the array

while (fscanf (output) != EOF)

{ fprintf (output, "%1.2f"); // start inputting it output  
i++; }

fclose (input1); // close files

fclose (input2);

len = i; // sets length as i

for (i=0; i < len; i++) // reads full array

{ for (j=i+1; j < len; j++) // reads individual values

{ if (arr3[i] > arr3[j]) // compares array values

{ temp = arr3[i];

arr3[i] = arr3[j];

arr3[j] = temp;

}

}

}

// print each value

for (i=0; i < len; i++)

{ fprintf (output, "%1.2f");

}

// computes sum and average

for (i=0; i < len; i++)

{

sum = arr3[i] + arr3[i+1];

sum += sum;

i++;

avg = sum / i;

}

Tyler Hansen 115064679

// print to output file

fprintf(output, "sum is: %.f", sum);

fprintf(output, "In average is: %.f", avg);

fclose(output); // close output

}

return 0;

}

Q 4:

Print array!

-3





Q2: #include &lt;stdio.h&gt;

int main()

{ int i, n, t<sub>1</sub>, t<sub>2</sub>, t<sub>3</sub>; and t are used for increments  
and next values. n is for iterations

float x, sum, sum2 // x is input value, sum is final value  
printf("Enter value for x: "); // value for computation  
scanf("%f", &x);

printf("Enter value for n: "); // value for iterations  
scanf("%d", &n);

i = 0; // starts i as 0

t = x - 1; // first t value is x - 1

sum = t; // sets first value for sum as t initial

for (i = 0; i < n; i++) // counts until n iterations

{ // t is what we add on to  
t<sub>1</sub> = t<sub>1</sub> \* (-1 \* x \* x) / ((2 \* i + 2) \* (2 \* i + 3)); -5

sum = sum + t<sub>1</sub>; // stores sum as the value of t and  
adds it to the original sum

i++; // increments i to get to n

}

printf("ln(%.f) = %.f for %d iterations", x, sum, n);  
// printf above shows original x value, sum, and  
for how many iterations

// compute 3 smallest sum

for (i=0; i < n; i++) // goes for 3 values

{ // take from previous to have the sum

sum = t<sub>1</sub>; // finds the 3 values

t<sub>2</sub> = sum + t<sub>1</sub>;

t<sub>3</sub> = sum + t<sub>2</sub>;

i++;

if (t<sub>1</sub> < t<sub>2</sub> < t<sub>3</sub>) // sets three low values

{

sum2 = t<sub>1</sub> + t<sub>2</sub> + t<sub>3</sub>

}

-12

}

printf("smallest sum is: %.f", sum2);

return 0;

}

```

Q3: #include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define Max 32

int main()
{
    File *input1, *input2, *output // sets files
    fopen(input1, "input_sentence.txt", "r");
    fopen(input2, "names.txt", "r"); // opens all files
    fopen(output, "output_sentence.txt", "w");

    if (input1 == NULL || input2 == NULL || output == NULL)
    {
        /* if files == NULL, we can't open them
           i.e. the file is not there */
        printf("cannot open file.");
        exit(1);
    }

    char arr1[Max], arr2[Max], temp[Max]; // sets arrays
    int i=0, j=0 // sets i=0 (first value)
    // j=0 (used for specific values)

    while ((fscanf(input2, "%s", arr2)) != EOF) // looks in file 2
    {
        if (i==0) // check first character in string to see names
        {
            arr1[i] = (char)(int(arr2[i] - 32)); // make capital
            temp[j] = arr1[i];
        }
        else
        {
            temp[j] = arr2[i];
            j++;
            i++;
        }
    }

    temp[j] = '\0' // \0 is null

```

```
c = cgetfc(input) // output the phrase
```

```
if (c == ' ')
```

```
{  
    fprintf(output, " ", c)  
}
```

```
else if (c == '\n')
```

```
{  
    fprintf(output, "\n", c)  
}
```

```
else if (c == '\t')
```

```
{  
    fprintf(output, "\t", c)  
}
```

```
fclose(output)
```

```
}
```

```
return 0;
```

```
}
```

## ESE 124 Fall 2023 Midterm

ID: 115037528

20.5

First name Last name

Anthony Aguirre

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- a) Read Data (10 points)  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- b) Store Data as 1D Array (10 points)  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- c) Using Bubble Sort (10 points)  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- d) Write the Solution to Output File (5 points)  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

---

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- a) Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

---

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

---

**Q4: Bonus 10 points**

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
 Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

1150375 28

Anthony Aguirre

```
1) #include <stdio.h>
#include <stdlib.h>
int main()
```

```
FILE * I_File, In_File;
int F_File, S_File, N_File, Sort;
```

```
printf("Enter numbers for first array: \n"); // Create an array
scanf("%d", &a[F_File]);
```

```
printf("Enter numbers for second array: \n");
scanf("%d", &a[S_File]);
```

```
if (fopen("I_File", "input 1.txt", "r") != EOF); // read values from txt documents
else (fopen == NULL) {
    printf("Error can not open"); }
```

```
if (fopen("In_File", "input 2.txt", "r") != EOF);
else (fopen == NULL) {
    printf("Error can not open"); }
```

```
for ("In_File" & "input 2.txt"); // stores both arrays into one document
    scanf("%d", &a[N_File]);
```

```
for (i=0, i<0, i++) {
    (!sort);
```

```
fclose("input 1.txt");
```

```
fclose("input 2.txt");
```

```
return 0;
```

```
}
```

Q1: -26-4





115037528

Anthony Aguirre

2) #include <stdio.h>

```
int main() {
```

```
    int n, i;
```

```
    float sum, x;
```

```
    printf("Enter value for x: ");
```

```
    scanf("%f", &x);
```

```
    printf("Enter a precision value for n: ");
```

```
    scanf("%d", &n);
```

```
    sum = x;
```

```
    sum += i;
```

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

```
        sum = (-1)i * (x - 1)i * (i + 1) / (i + 2);
```

```
    }
```

```
    printf("Sum of ln(x) is %f", sum);
```

```
    return 0;
```

```
}
```

III Create values for x

III Create values for n.

III Creating the loop for ln(x)

III Ratio of ln(x)

-5

2) Part 20

# include <stdio.h>

```
int main() {
    int n, t;
    float x, sum = 0, N-sum = 0;

    printf("Enter value for x: \n");
    scanf("%f", &x);
    printf("Enter precision value for n: \n");
    scanf("%d", &n);

    t = x;
    sum = t;

    for (k = 0; k < n; k++) {
        t = (-1) * t * (x - 1) * (k + 1) / (k + 2);
        N-sum = N-sum + t;
    }

    printf("The smallest sum of n terms for ln(x) is %f", N-sum);

    return 0;
}
```

||| Entering value for x

||| Entering value for n.

||| Ratio for ln(x)

115037528

Antony Aguirre

3

```
#include <stdio.h>
```

```
int main()
```

```
{  
    FILE * I_File, * O_File;
```

```
    int In_File, Out_File, Name[30], N_File
```

```
    if (fopen("In_File", "input_Sentence.txt", "r") != EOF);
```

||| Reads  
what is in  
txt

```
    else (fopen == NULL) {
```

```
        printf("Error can not open");
```

```
    } if (fopen("N_File", "names.txt", "r") != EOF) {
```

||| Reads the  
name within  
the file.

```
        fgets(Name, sizeof(Name), stdin);
```

```
        printf("Name within the file is: \n");
```

```
    } if (fopen("O_File", "
```

-30



115037528

Anthony Aguirre

Bonus

```
#include <stdio.h>
```

```
int main() {
```

```
    int num, N-sum, x
```

```
    printf("Enter a value for x: ");
```

```
    scanf("%d", &x);
```

+0.5





## ESE 124 Fall 2023 Midterm

ID: 114982059

70

First name Last name

James Bondino

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array.** The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- Read Data (10 points)**  
Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.
- Store Data as 1D Array (10 points)**  
Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.
- Using Bubble Sort (10 points)**  
Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.
- Write the Solution to Output File (5 points)**  
Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."  
Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- Define the variables and initial them (5 points).

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

---

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**rank are taking an exam.*

---

**Q4: Bonus 10 points**

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.



11/09/2019,

Times Bandwidth

Q1

Q1:-2

#include &lt;stdio.h&gt;

void merge\_sorted\_arrays (double arr1[], double arr2[], int size1, double result[], int size3)

{  
int i=0, j=0, k=0

while (i &lt; size1 &amp;&amp; j &lt; size2)

{  
if (arr1[i] < arr2[j]){  
result[k++] = arr1[i++];

else if (arr1[i] &gt; arr2[j])

{  
result[k++] = arr2[j++];

else

{  
i++;  
j++;

}

}  
while (i < size1){  
result[k++] = arr1[i++];

while (j &lt; size2)

{  
result[k++] = arr2[j++];}  
size3 = k;

{

// declare function  
merge

// declare variables

// if statements  
that sort two arrays

// increment array i, and j.

// merge arrays

void findSum ()

// declare sum function

```
{
    int maxStart = 0, maxEnd = 0;
    double maxSum = 0;
    int currentStart = 0;
    double currentSum = 0;
    double num;
    int index = 0;
```

// find max continuous sum

// declare max

while ( fscanf(<sup>output</sup> ~~input~~ file, "%lf", &num) == 1)

-2

```
{
    if (currentSum + num >= num)
    {
        currentSum += num;
    }
    else
```

// sum + num > num  
then add together

```
{
    currentSum = num;
    currentStart = index;
}
if (currentSum > maxSum) {
    maxSum = currentSum;
    maxStart = currentStart;
    maxEnd = index;
}
index++;
}
```

// if num + sum > max sum  
then maxSum = current sum

// increment

int main () {

```
    file *inputFile1, *inputFile2, *output input file;
    inputFile1 = fopen("input1.txt", "r");
    inputFile2 = fopen("input2.txt", "r");
    outputFile = fopen("output.txt", "w");
    if (inputFile1 == null || inputFile2 == null || outputFile == null) {
        printf("error opening files\n");
        return 1;
    }
}
```

// open files

~~should be in front~~

// if any  
can't open  
-1

Q1 continued

James Berhno, 114982059

```
double arr1[30], arr2[30], result[60];  
int size1 = 0, size2 = 0, size3 = 0;
```

```
while (fscanf(inputfile1, "%d", &arr1[size1]) == 1) { // enter arr1  
    size1++;  
}
```

```
while (fscanf(inputfile2, "%d", &arr2[size2]) == 1) { // enter arr2  
    size2++;  
}
```

```
mergeSortedArray(arr1, arr2, size1, size2, result, &size3);  
for (int i = 0; i < size3; i++)  
{  
    fprintf(outputfile, "%d", result[i]);  
}
```

// merge array

// display result

```
findSum();
```

// find sum

```
fprintf(outputfile, "largest Sum", maxSum);  
float float average = maxSum / size3;
```

// average formula

```
fprintf("average", maxSum / size3, "%d", average);
```

// display avg

```
fclose(inputfile1);
```

```
fclose(inputfile2);
```

```
fclose(outputfile);
```

// close files

```
return 0;
```

```
}
```

// exit program



Q2: Taylor series of  $\ln(x)$

Times Borno 114782059

```
#include <stdio.h> // define libraries
#include <math.h>
```

```
int main()
```

```
{
    double x, a, term; sum; // declare variables
    int k=1;
```

```
    printf("enter x", x); // declare and store variables
    scanf("%lf", &x);      x and a
```

```
    printf("enter precision", a);
    scanf("%lf", &a);
```

```
    term = (x-1)/x; // intro eq = (x-1)/x
    sum = term;      // sum = term
```

```
    while (fabs(term) >= a)
```

```
    {
        k++
```

```
        // increment k
```

```
        term = -term
```

```
        sum += term
```

```
        // term = term * term or (x-1)/x + 1
        // add the terms up
```

```
    }
    printf("ln(%.2lf) = %.10lf", x, sum);
```

```
    return 0;
```

```
}
```

```
// print value of
// ln(x) and its sum.
```

```
// exit code
```

-5

Part 2:

$\ln(x)$

```
#include <stdio.h>
```

```
#include <math.h>
```

// declare  
libraries

```
int main()
```

```
{
```

```
int x1, x2, x3;
```

// declare  
variables

```
int term;
```



$\ln(0) = \text{undefined}$

~~$\ln(x_1) + \ln(x_2) + \ln(x_3)$~~

```
printf("Enter x1, x2, x3, x1, x2, x3");  
scanf("%d %d %d", &x1, &x2, &x3);
```

```
x1 = 1;
```

```
x2 = 2;
```

```
x3 = 3;
```

// set x = value

```
int t1, t2, t3;
```

```
t1 = ln(x1);
```

```
t2 = ln(x2);
```

```
t3 = ln(x3);
```

// find  $\ln(x_1, x_2, x_3)$

```
float sum;
```

// declare sum variable

```
sum = t1 + t2 + t3;
```

```
printf("sum", "%lf", sum);
```

// find and print sum

```
return 0;
```

-13

Q3:

James Berdno

114987059

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
```

```
int main ()
```

```
{
```

```
File *input_sentence.txt, *names.txt
```

```
input file 1 = fopen("input_sentence.txt", "r");
```

```
input file 2 = fopen("names.txt", "r");
```

// opens and  
reads file

```
Output file = fopen("output_sentence.txt", "w");
```

```
if (input file 1 == null || input file 2 == null || output file == null)
```

```
{
    printf("Error opening file");
    return 1;
}
```

// if files = null  
then exit code

```
int char;
```

```
int capitalize (char *line) {
    char *token = strtok(line, " ");
    while (token != null) {
        int len = strlen(token);
```

// capitalize

-10

~~strtok;~~  
copy:

```
Verf merge files ("output", result);
```

// merge files from 2  
function

```
}
```

```
printf("output file", %c, output file);
```

```
fclose(input file 1);
```

// print new output file

```
fclose(input file 2);
```

// close files

```
fclose(output file);
```

```
return 0;
```

```
}
```

// exit code

Q2: Taylor series of  $\ln(x)$

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

int main()
{
    double x, a, term, sum;
    int k = 1;
    printf("Enter a desired precision: ");
    scanf("%lf", &a);
    term = (x - 1) / x;
    sum = term;
    while (fabs(term) >= a)
    {
        k++;
        term = term * term;
        sum += term;
    }
    printf("ln(%.1f) = %.10lf\n", x, sum);
    return 0;
}
```



## ESE 124 Fall 2023 Midterm

ID: 114909468

18

First name Last name

Arrelis Villa-Perez

**Q1: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Write a C program that merges two 1D arrays into a third **sorted 1 D array that contains no duplicates, then computes and displays the sum and average of the values in the sorted array**. The two 1 D arrays can have each at most 30 decimal values and are read from two separate input files. The third 1 D array is written in an output file.

**Input example:**

First 1D array 2.0 4.0 3.0 2.0

second 1D array is 8.0 5.0 7.0 2.0

**Output file:**

The third 1D array (result) is 2.0 3.0 4.0 5.0 7.0 8.0

Sum is: 29.0

Average is: 4.83

**Steps:**

- a) Read Data (10 points)

Write a C program that reads decimal numbers from two input files, "input1.txt" and "input2.txt," and stores them in separate arrays.

- b) Store Data as 1D Array (10 points)

Modify your program to store the decimal numbers from both files into a single 1D array with no duplicates. Display the contents of the 1D array.

- c) Using Bubble Sort (10 points)

Implement the bubble sort algorithm to sort the decimal numbers in ascending order within the 1D array. Display the sorted array.

- d) Write the Solution to Output File (5 points)

Write the sorted array, sum of the decimal numbers, and average of the decimal numbers to an output file named "output.txt."

Ensure that the output file is properly formatted and includes appropriate labels for each value.

**Q2: 30 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Part 1: Design a C program to compute and display the  $n$  terms of the logarithm function  $\ln(x)$  as well as their sum:

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 \dots$$

The decimal value  $x$  and the integer value  $n$  are read from the keyboard.

- a) Define the variables and initial them (5 points).

$$(x-1)^1 - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \frac{(x-1)^5}{5} - \frac{(x-1)^6}{6} + \frac{(x-1)^7}{7}$$

- b) Find the next term related to the previous term(5 points).
- c) Sum to the terms by using a loop. (5 points)

Part 2: Find the three consecutive terms  $t_k, t_{k+1}, t_{k+2}$  [ among the  $n$  terms of the  $\ln(x)$  series ] that have the smallest sum  $t_k + t_{k+1} + t_{k+2}$ .

**Q3: 35 points** (Use comments in your code to explain your solution, including algorithm - or you will lose points)

Design a C program that performs the following tasks:  
 Read words from an input file named "input\_sentence.txt".  
 Read names from an input file named "names.txt".  
 Create a new file named "output\_sentence.txt".  
 Write the sentence from "input\_sentence.txt" into "output\_sentence.txt", but with the first letter of each word appearing in "names.txt" capitalized.

For example, given the following input files:

**input\_sentence.txt:**

*Hi, james and frank are taking an exam.*

**names.txt:**

*james, frank, chris, anne*

The program should generate the following output file:

**output\_sentence.txt:**

*Hi, **J**ames and **F**ranks are taking an exam.*

**Q4: Bonus 10 points**

Data is provided in an 8 bits variable called par, The bits[7, 3] are data and bits[2, 0] are checksum. (eg: 11010011, 1+1+0+1+0=3).

Write a program that verifies the checksum.

- a) Isolate bits [7-3] eg: 11010  
 Isolate bits [2-0] eg: 011
- b) Calculate checksum value of bits [7, 3] by using a loop
- c) Verify only using bitwise operators.

11490946

File Villa - pr

Q1

```
#include <stdio.h>
```

```
#include <math.h>
```

```
File * Inp_F = fopen("input1.txt", "r") // read
```

```
File * Inp_F2 = fopen("input2.txt", "r") // read
```

```
File * Out_F = fopen("output.txt", "w") // write
```

```
if (Inp_F == NULL || Inp_F2 == NULL || Out_F == NULL)
```

```
{  
    printf("Error: cannot open files\n"); // Error handle  
    exit(1);  
}
```

```
int array1[MaxLen], array2[MaxLen];
```

```
while (fscanf(Inp_F, "%d %d %d %d", &array1[i]) != EOF) // read &  
    printf("%d %d %d %d", &array1[i]); // read &  
    // store data
```

```
while (fscanf(Inp_F2, "%d %d %d %d", &array2[i]) != EOF)  
    printf("%d %d %d %d", &array2[i]);
```



Q2

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

```
int main () {
```

```
    int i
```

```
    float x, a, sum, t;
```

```
    printf ("Enter Value for x: "); // decimal value
```

```
    scanf ("%f", &x);
```

```
    printf ("Enter value for n: "); // integer value
```

```
    scanf ("%f", &a);
```

```
    t = x - 1 // Initial value
```

```
    sum = t;
```

```
    i = 0;
```

```
    while (fabs(t) > a);
```

```
    t = t - (x - 1)2 / 2
```

- 8 + 3

```
    sum += t;
```

```
    i++;
```

P2: -15

```
}
```

```
    printf ("find sum for ln(1+f) = %.f", x, sum);
```

```
    return 0;
```

```
}
```

Q2: -23



114909408  
Arelis Vill

Q3

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

```
File *inp-F = fopen("input_sentence.txt", "r"); //read
File *inp-F2 = fopen("name.txt", "r"); //read
File *out-F = fopen("output_sentence.txt", "w"); //write

if (inp-F == NULL || inp-F2 == NULL || out-F == NULL)
{
    printf("Error: cannot open file.\n"); // Error handle
    exit(1);
}
```

```
char a = sentence;
char b = name;
```

```
while (fscanf(inp-F, "%c", &a) != EOF)
{
    printf("%c", &a[i]); // store data
}
```

```
while (fscanf(inp-F2, "%c %c %c", &b) != EOF)
{
    printf("%c %c %c", &b);
}
```

```
for (char a; char a + char b,
```

-29

Return 0;





114909468

Arelis Villa

Q4.

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main () {
```

