SEARCHING ALGORITHMS

1. Refer the code in “Binary Search” in socodery folder. Modify the existing implementation to include the following operations.
a. Read lines instead of words
b. Sort the remaining lines
c. Search for a line and if found delete the line
d. Final list of lines after above operations to be written to a file out.txt
Add required functions
Test with dataset below
Line 9- 90 words
Line 2 - 89 words
Line 7-45 words
Line 3 -45 words
Line 4 -45 words
Line 3 -45 words

Sol:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_LINES 1000

#define MAX_LENGTH 1000


int readLinesFromFile(const char *filename, char lines[MAX_LINES][MAX_LENGTH]) {

  FILE *file = fopen(filename, "r");

  if (!file) {

    printf("Unable to open file: %s\n", filename);

    return 0;

  }


  int i = 0;

  while (fgets(lines[i], MAX_LENGTH, file)) {

    lines[i][strcspn(lines[i], "\n")] = '\0';

    i++;

  }
```

```
    fclose(file);
    return i;
}

void writeLinesToFile(const char *filename, char lines[MAX_LINES][MAX_LENGTH], int
numLines) {
    FILE *file = fopen(filename, "w");
    if (!file) {
        printf("Unable to open file: %s\n", filename);
        return;
    }

    for (int i = 0; i < numLines; i++) {
        fprintf(file, "%s\n", lines[i]);
    }

    fclose(file);
}

int compareLines(const void *a, const void *b) {
    return strcmp(*(const char **)a, *(const char **)b);
}

int binarySearch(char lines[MAX_LINES][MAX_LENGTH], int low, int high, const char *target) {
    while (low <= high) {
        int mid = low + (high - low) / 2;
        int result = strcmp(lines[mid], target);
        if (result == 0) {
            return mid;
        } else if (result < 0) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}

void deleteLine(char lines[MAX_LINES][MAX_LENGTH], int *numLines, const char *target) {
    int index = binarySearch(lines, 0, *numLines - 1, target);
    if (index != -1) {
        for (int i = index; i < *numLines - 1; i++) {
            strcpy(lines[i], lines[i + 1]);
        }
        (*numLines)--;
        printf("Line deleted: %s\n", target);
    } else {
        printf("Line not found: %s\n", target);
    }
}

int main() {
    char lines[MAX_LINES][MAX_LENGTH];
    int numLines;
    char targetLine[MAX_LENGTH];
    char inputFilename[] = "input.txt";
    char outputFilename[] = "out.txt";

    numLines = readLinesFromFile(inputFilename, lines);

    if (numLines == 0) {
        printf("No lines to process.\n");
        return 1;
    }

    qsort(lines, numLines, sizeof(lines[0]), compareLines);

    printf("Enter the line to search and delete: ");
    fgets(targetLine, MAX_LENGTH, stdin);
    targetLine[strcspn(targetLine, "\n")] = '\0';

    deleteLine(lines, &numLines, targetLine);

    writeLinesToFile(outputFilename, lines, numLines);

    printf("Final lines written to %s\n", outputFilename);

    return 0;
}
```

Output:

```
Line 2 - 89 words
Line 3 - 45 words
Line 4 - 45 words
Line 7 - 45 words
Line 9 - 90 words
```