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

// Question-1
// ----------
int negative(int n)
{
    if (n < 0)
    {
        return 1;
    }
    else
        return -1;
}
void swap(int *a, int *b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
int partition(int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++)
    {

        if (arr[j] < pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {

        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

// Question-2
// -----------

void para(char a[5000])
```

```
{
    int i = 0;
    while (a[i] != '\0')
    {
        if (a[i] == ' ' || a[i] == ',' || a[i] == '.')
        {
            printf("\n");
        }
        else
        {
            printf("%c", a[i]);
        }
        i++;
    }
}
// void word(char a[500][50])
// {
//     int str[500][50];
//     int j, i = 0;
//     int c = 0;
//     for (i = 0; i < 500; i++)
//     {
//         for (j = 0; j < 50; j++)
//         {
//             if (a[c++] == ' ' || a[c++] == '.' || a[c++] == ',')
//             {
//                 str[i][j] = '\0';
//                 break;
//             }
//             else
//             {
//                 str[i][j] = a[c++];
//             }
//         }
//     }
// }
// Question-3
// -----------
void file1(int n, int a[1000])
{
    int i;
    srand((unsigned int)time(NULL));

    n = rand() % 69 + 10;
    for (i = 0; i < 1000; i++)
    {
        n = rand() % 69 + 10;
        a[i] = n;
    }
}
void identify(int a[1000], int b[1000])
{
    file1(1000, a);
    for (int i = 0; i < 1000; i++)
    {
        if (a[i] > 15 && a[i] < 47)
        {
```

```
            b[i] = a[i];
        }
    }
}
int main()
{
    int c = 0, d = 0;

    int i, j, k = 0, l = 0;
    int a[] = {-7, 3, 80, 12, -35, 28, -54, 61, 12, 3, 9, -5, 35, 52,
-96, 29, -12, 27, 37, -42, 53, 48, 63, -51, -75, 19, -11, 2,
               -81, 55, -14, 41, -29, 97, -245};
    for (i = 0; i < 35; i++)
    {
        if (negative(a[i]) == 1)
        {
            c++;
        }
        else
        {
            d++;
        }
    }
    int b[c], p[d];
    for (j = 0; j < 35; j++)
    {
        if (negative(a[j]) == 1)
        {
            b[k] = a[j];
            k++;
        }
        else
        {
            p[l] = a[j];
            l++;
        }
    }
    quickSort(b, 0, c - 1);
    quickSort(p, 0, d - 1);
    printf("The negative numbers are in increasing order\n");
    for (j = 0; j < c; j++)
    {
        printf("%d ", b[j]);
    }
    printf("\n");
    printf("The positive numbers are in decreasing order\n");
    for (j = d - 1; j >= 0; j--)
    {
        printf("%d ", p[j]);
    }
    char abc[] = "A stochastic fractal is built out of probabilities
and randomness. It is statistically self-similar. We\
will look at both deterministic and stochastic techniques for
generating fractal patterns. A line is\
self-similar. A line looks the same at any scale, but it’s not a
fractal. A fractal is characterized\
by having a fine structure at small scales, you’ll continue to find
```

```
fluctuations, and cannot be\
described with Euclidean geometry. If you can say, it’s a line, then
it’s not a fractal. Another\
fundamental component of fractal geometry is recursion. Fractal has a
recursive definition.\
We’ll start with recursion before developing techniques and code
examples for building fractal";
    printf("\n");
    printf("The words in different lines\n");
    para(abc);
    /*word(abc);
    printf("%s\n", abc);*/
    int arr[1000];
    int m = 0;
    file1(1000, arr);
    FILE *f;
    int arr1[1000];
    f = fopen("163-input.txt", "w+");
    if (f == NULL)
    {
        printf("Cannot open the file\n");
    }
    else
    {
        for (int g = 0; g < 1000; g++)
        {
            fprintf(f, " % d", arr[g]);
        }
    }
    fclose(f);
    identify(arr, arr1);
    quickSort(arr1, 0, 999);
    FILE *f1;

    f1 = fopen("163-output.txt", "w+");
    if (f1 == NULL)
    {
        printf("Cannot open the file\n");
    }
    else
    {
        for (int g = 999; g >= 0; g--)
        {
            fprintf(f, " % d", arr1[g]);
        }
    }
    fclose(f1);

    return 0;
}
```