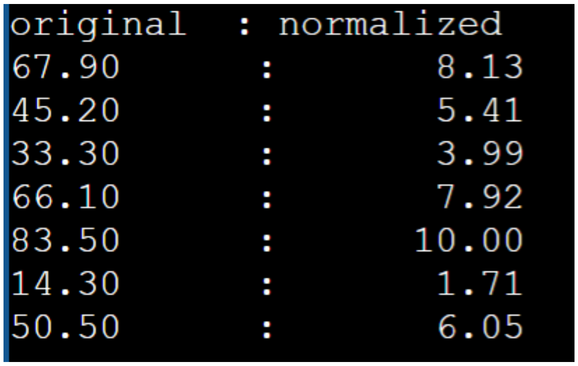
1.

```
#include <stdio.h>


int main()

{

    // vars

    /*

    xhigh = highest normalized value

    xLow = smallest normalized value


    xLow and xhigh have to be foudn throgh for loop (initialize the values as random data set numbers)


    */

    double min = 0, MAX = 0, xH, xL, normxi;

    int size=0;


    // 1. obtain files ////////////////////////////////////////////////

    // read from file

    FILE * file=fopen("data.txt", "r");


    // get size

    fscanf(file, "%d", &size);


    double array[size];


    // get min and max

    fscanf(file, "%lf", &min);

    fscanf(file, "%lf", &MAX);


    // get rest of values & find xL and xH

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

```
    fscanf(file, "%lf", &array[i]);

    if (i==0){
        xL == array[i];
        xH == array[i];
    }

    // sorting for xl and xH
    if (xL > array[i]){
        xL = array[i];
    }
    if (xH < array[i]){
        xH = array[i];
    }

}

// 2. normalize //////////////////////////////////////////
double nValues[size];

for (int i=0; i<size; i++){
    nValues[i] = min + (array[i] - xL) * (MAX - min) / (xH -xL);
}

// 3. printf ///////////////////////////////////////////////////////////
//vars
char * a = "original";
char * b = "normalized";

// origial: normal
printf("%-10s: %10s", a, b);
```

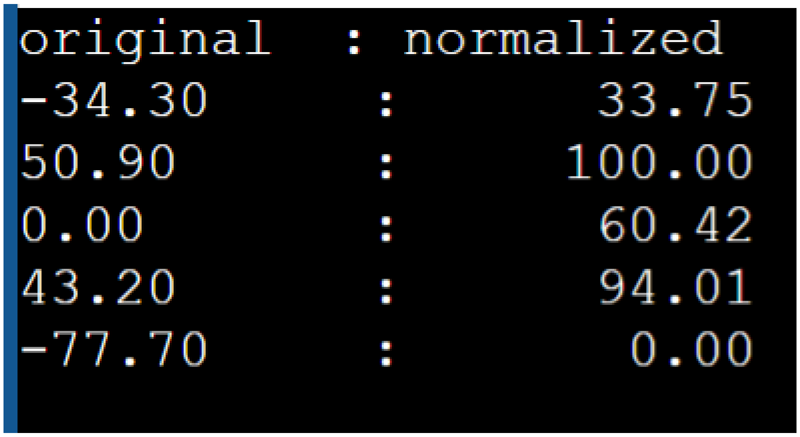
```
  for (int i=0; i<size; i++){
    printf("\n%-10.2lf : %10.2lf", array[i], nValues[i]);
  }

  // closing
  return 0;
}
```

1.



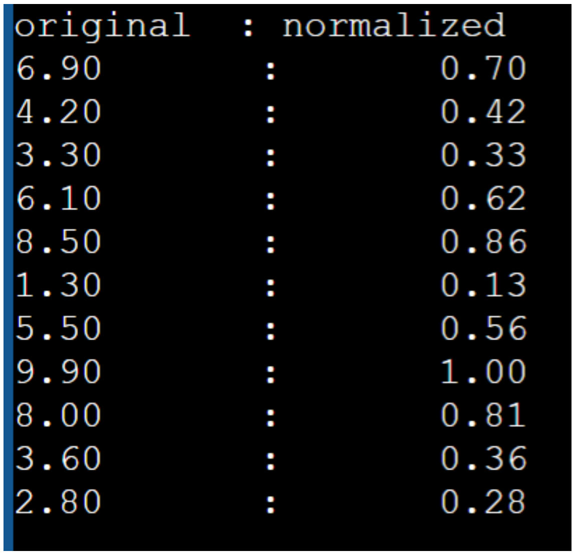
```
original  : normalized
67.90      :       8.13
45.20      :       5.41
33.30      :       3.99
66.10      :       7.92
83.50      :      10.00
14.30      :       1.71
50.50      :       6.05
```

2.



```
original  : normalized
-34.30     :      33.75
50.90      :     100.00
0.00       :      60.42
43.20      :      94.01
-77.70     :       0.00
```

3.

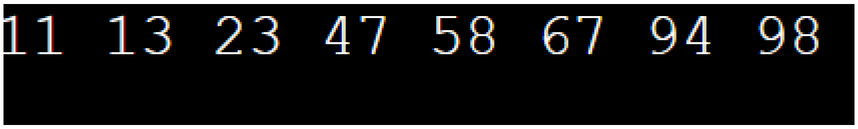


```
original  : normalized
6.90       :       0.70
4.20       :       0.42
3.30       :       0.33
6.10       :       0.62
8.50       :       0.86
1.30       :       0.13
5.50       :       0.56
9.90       :       1.00
8.00       :       0.81
3.60       :       0.36
2.80       :       0.28
```

2.

```
#include <stdio.h>
#define ARRAY_SIZE 8

// finds the position of the smallest element in the subarray
// list[first] through list[last].
// Pre: first < last and elements 0 through last of array list are defined.
// Post: Returns the subscript k of the smallest element in the subarray;
// i.e., list[k] <= list[i] for all i in the subarray

int get_min_range (int list[], int first, int last)
{
  // finding min
  int min = list[first], pos = first;

    for (int i=first; i<=last; i++){
        if (min > list[i]){
            min = list[i];
            pos = i;
        }
    }
    return pos;
}

// sorts the data in array list
void select_sort(int list[], int n)
{
     int fill,          /* index of first element in unsorted subarray    */
         temp,          /* temporary storage                              */
         index_of_min;  /* subscript of next smallest element             */
```

```
   for (fill = 0; fill < n-1; ++fill) {

      /* Find position of smallest element in unsorted subarray */

      index_of_min = get_min_range (list, fill, n-1);


      /* Exchange elements at fill and index_of_min */

      if (fill != index_of_min) {

          temp = list[index_of_min];

          list[index_of_min] = list[fill];

          list[fill] = temp;

      }

   }

}


int

main (void) {

  int array[] = {67, 98, 23, 11, 47, 13, 94, 58};

  int i;


  select_sort (array, ARRAY_SIZE);


  for (i=0; i < 8; ++i)

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


  return (0);

}
```

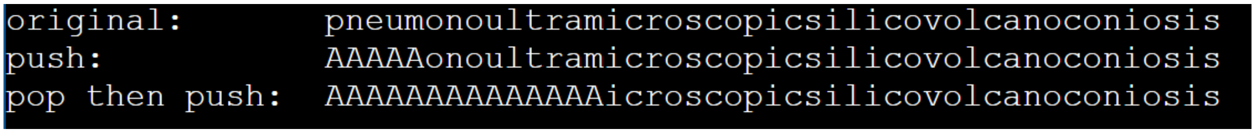


```
11 13 23 47 58 67 94 98
```

3.

```
#include <stdio.h>

#define STACK_EMPTY '0'

#define STACK_SIZE 200


void push(char stack[],    /* input/output - the stack */

     char item,       /* input - data being pushed onto the stack */

     int  *top,       /* input/output - pointer to top of stack */

     int  max_size)   /* input - maximum size of stack */

{

     if (*top < max_size-1) {

         ++(*top);

         stack[*top] = item;

     }

}


char pop (char stack[],    /* input/output - the stack */

    int *top)        /* input/output - pointer to top of stack */

{

    char item;       /* value popped off the stack */


    if (*top >= 0) {

        item = stack[*top];

        --(*top);

    } else {

         item = STACK_EMPTY;

    }


    return (item);

}
```

```
int
main (void)
{
  char s [STACK_SIZE] = "pneumonoultramicroscopicsilicovolcanoconiosis";
  char p = 'A';
  int s_top = -1; // stack is empty

   printf("%-15s %s\n", "original: ", s);
  for (int i=0; i<5;i++){
     push(s, p, &s_top, STACK_SIZE);
  }
  printf("%-15s %s\n", "push: ", s);

   for (int i=0; i<3;i++){
     pop(s, &s_top);
  }

  for (int i=8; i<20;i++){
     push(s, p, &s_top, STACK_SIZE);
  }
  printf("%-15s %s\n", "pop then push: ", s);

  return (0);
}
```



```
original:       pneumonoultramicroscopicsilicovolcanoconiosis
push:           AAAAAonoultramicroscopicsilicovolcanoconiosis
pop then push:  AAAAAAAAAAAAAicroscopicsilicovolcanoconiosis
```

Pxy