Harsh Solanki Lab 6 CPS188 section 16

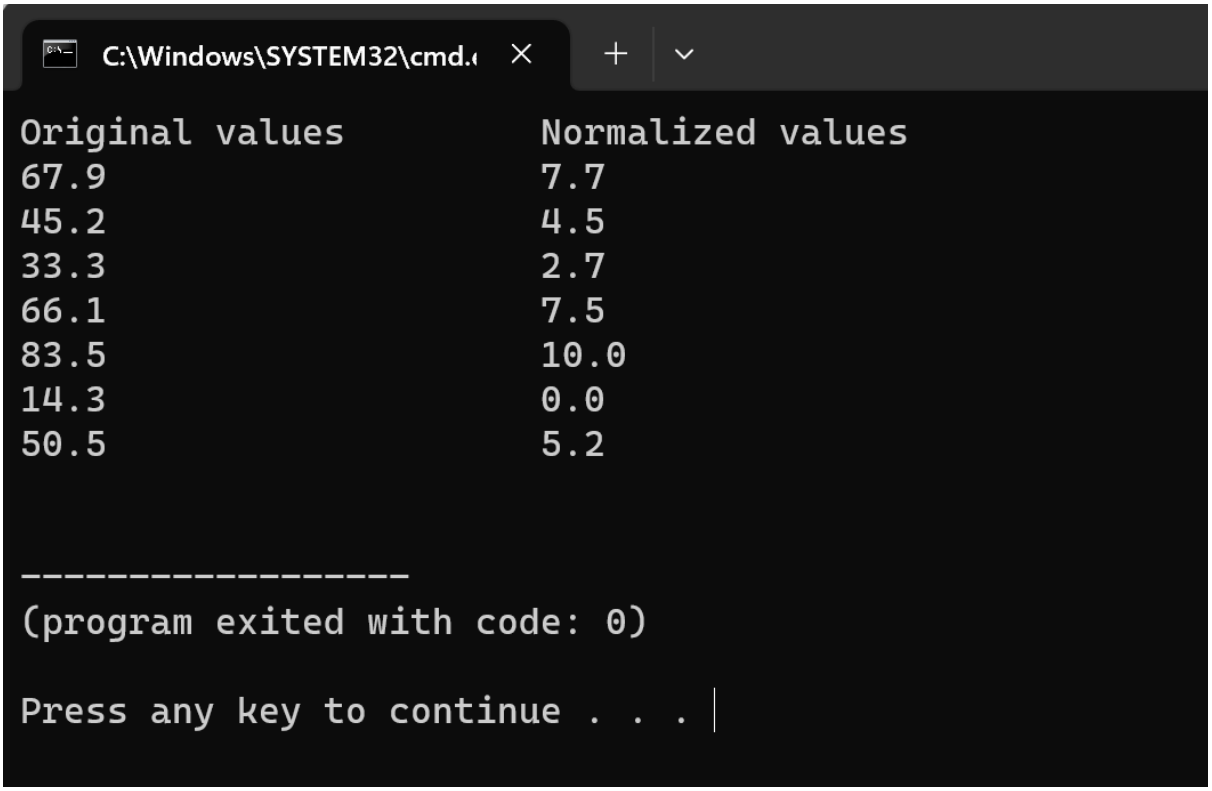
Problem 1:

Source Code:

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

double min(double x, double y)
{
    if(x > y) return y;
    return x;
}

double max(double x, double y)
{
    if(x < y) return y;
    return x;
}

int main()
{
    int counter;
    double minimum, maximum;
    FILE * fpointer = fopen("file.txt", "r");
    fscanf(fpointer, "%d %lf %lf", &counter, &minimum,
&maximum);
    double x[counter], xlow, xhigh;
    for(int i = 0; i < counter; i++)
    {
        fscanf(fpointer, "%lf", &x[i]);
        if(i == 0)
        {
            xlow = x[i];
            xhigh = x[i];
        }
        xlow = min(xlow, x[i]);
        xhigh = max(xhigh, x[i]);
    }
    double normx[counter];
    for(int i = 0; i < counter; i++){
```

```
        normx[i] = minimum + (x[i] - xlow) * (maximum -
minimum) / (xhigh - xlow);
    }
    printf("Original values         Normalized values\n");
    for(int i = 0; i < counter; i++)
    {
        printf("%.1f\t\t\t%.1f\n", x[i], normx[i]);
    }
    return 0;
}
```

Test 1, with values 7 0.0 10.0 67.9 45.2 33.3 66.1 83.5 14.3 50.5



Test 2 with values 11 0.0 1.0 6.9 4.2 3.3 6.1 8.5 1.3 5.5 9.9 8.0 3.6 2.8

```
C:\Windows\SYSTEM32\cmd.

Original values          Normalized values
6.9                      0.7
4.2                      0.3
3.3                      0.2
6.1                      0.6
8.5                      0.8
1.3                      0.0
5.5                      0.5
9.9                      1.0
8.0                      0.8
3.6                      0.3
2.8                      0.2


------------------
(program exited with code: 0)

Press any key to continue . . .
```

Test 2 with values 5 0.0 100.0 -34.3 50.9 0.0 43.2 -77.7

```
C:\Windows\SYSTEM32\cmd.

Original values          Normalized values
-34.3                    33.7
50.9                     100.0
0.0                      60.4
43.2                     94.0
-77.7                    0.0


------------------
(program exited with code: 0)

Press any key to continue . . .
```

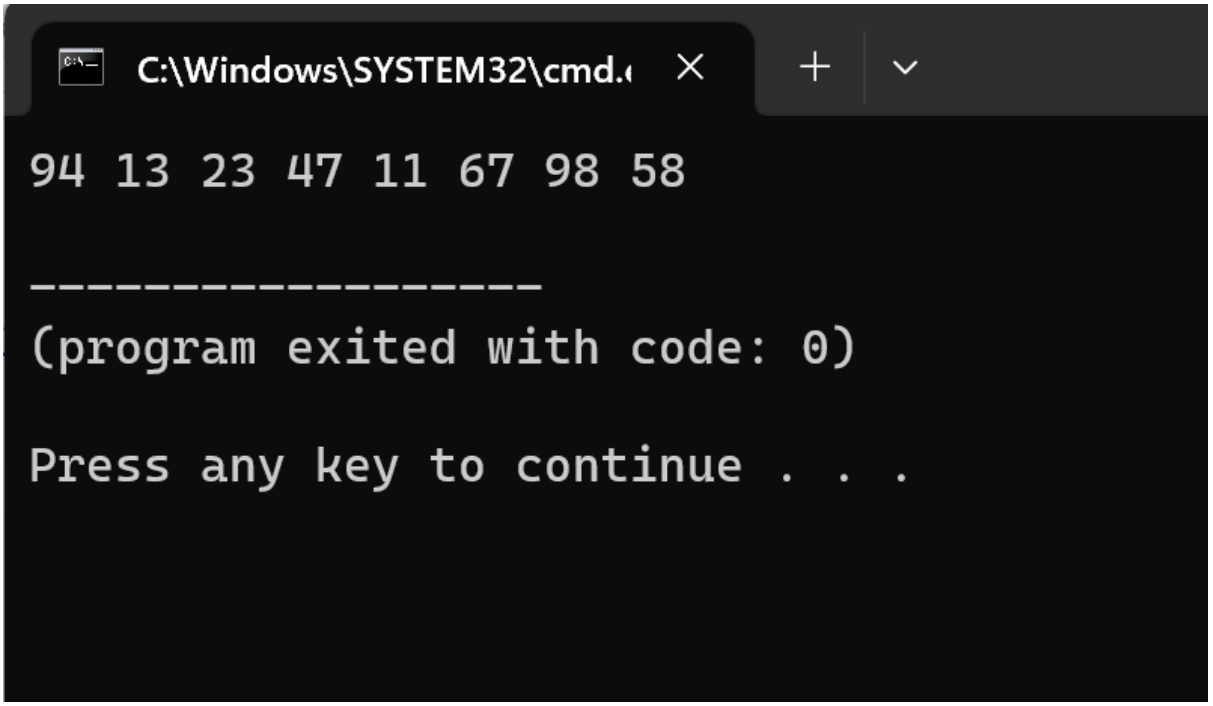
Problem 2:

Source code:

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

int get_min_range (int list[], int first, int last)
{
   int lowest = list[first];
   for (int i = first; i<=last; i++)
   {
        if (list [i] < lowest)
        {
             lowest = list[i];
        }
   }
     for (int x = first; x<last; x++)
     {
          if (list[x] == lowest)
          {
               return x;
          }
     }
     return 0;
}

void
select_sort(int list[], int n)
{
     int fill, temp,index_of_min;

     for (fill = 0; fill < n-1; ++fill)
     {
          index_of_min = get_min_range (list, fill, n-1);
          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);
}
```

Test Run:



Problem 3:

Source code:

```
#include <stdio.h>
#define STACK_EMPTY '0'
#define STACK_SIZE 20
```

```
void
push(char stack[],char item,int  *top,int  max_size)
{
     if (*top < max_size-1)
     {
         ++(*top);
         stack[*top] = item;
     }
}

char pop (char stack[],int *top)
{
    char item;

    if (*top >= 0)
    {
        item = stack[*top];
        --(*top);
    }
    else
    {
         item = STACK_EMPTY;
    }

    return (item);
}

int main (void)
{
   char s [STACK_SIZE];
   int s_top = -1;

   /* complete the program here */

   push(s, 'a', &s_top, STACK_SIZE);
   push(s, 'b', &s_top, STACK_SIZE);
   push(s, 'c', &s_top, STACK_SIZE);

     for (int i=0; i<s_top+1; i++)
     {
          printf("%c", s[i]);
     }

     printf("\n\n");
```

```
    printf("pop removed: %c\n", pop(s, &s_top));

    for(int i=0; i<s_top+1; i++)
    {
        printf("%c", s[i]);
    }
  return (0);
}
```

Test run: