

# *CPS 188 Lab 3 : Arrays*

*Instructor: Dr. Ufkes*

*TA: Mohammed Emrul Hasan*

*Section: 18*

*Sayeed Ahamad*

*Student Number: 501209136*



# 1 Problem Sets

## 1.1 Problem 1

### 1.1.1 Computer Program

```
1 #include <stdio.h>
2
3 void main(void)
4 {
5     FILE* file;
6     int amt, i;
7     double min, max, x1, xh;
8     file = fopen("data.txt", "r");
9     while (!feof(file))
10    {
11        fscanf(file, "%d %lf %lf", &amt, &min, &max);
12        double nums[amt], norm[amt];
13        for (int i = 0; i < amt; i++)
14        {
15            fscanf(file, "%lf", &nums[i]);
16        }
17        x1 = nums[0];
18        xh = nums[1];
19        for (int i = 0; i < amt; i++)
20        {
21            if (nums[i] < x1)
22            {
23                x1 = nums[i];
24            }
25            else if (nums[i] > xh)
26            {
27                xh = nums[i];
28            }
29        }
30        for (int i = 0; i < amt; i++)
31        {
32            norm[i] = min + ((nums[i] - xh) * (max - min) / (
xh - x1));
33        }
34        printf("Original\t\tNormalized\n");
35        printf("-----\t\t-----\n");
36        for (int i = 0; i < amt; i++)
37        {
38            printf("%.1lf\t\t%.1lf\n", nums[i], norm[i]);
```

```

39     }
40     fclose(file);
41 }
42 }

```

Listing 1.1: *Program to print the values of the Pascal's Triangle in Sequential Order*

## 1.1.2 Program Output Screenshot

```

source Code/Lab 4$ ./pascal
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1

```

## 1.2 Problem 2

### 1.2.1 Computer Program

```

1
2
3 #include <stdio.h>
4 #define ARRAY_SIZE 8
5 // finds the position of the smallest element in the
  subarray
6 // list[first] through list[last].
7 // Pre: first < last and elements 0 through last of array
  list are defined.
8 // Post: Returns the subscript k of the smallest element in
  the subarray;
9 // i.e., list[k] <= list[i] for all i in the subarray
10 int get_min_range (int list[], int first, int last)
11 {
12     int j;
13     int min = sizeof(double);
14     for (int i = 0; i < last; i++)

```

```

15     {
16         if (min > list[i])
17         {
18             min = list[i];
19             j = i;
20         }
21     }
22     return j;
23 }
24 // sorts the data in array list
25 void select_sort(int list[], int n)
26 {
27     int fill, /* index of first element in unsorted
28 subarray */
29     temp, /* temporary storage */
30     index_of_min; /* subscript of next smallest element
31 */
32     for (fill = 0; fill < n-1; ++fill)
33     {
34         /* Find position of smallest element in unsorted
35 subarray */
36         index_of_min = get_min_range (list, fill, n-1);
37         /* Exchange elements at fill and index_of_min */
38         if (fill != index_of_min)
39         {
40             temp = list[index_of_min];
41             list[index_of_min] = list[fill];
42             list[fill] = temp;
43         }
44     }
45 }
46 int main (void)
47 {
48     int array[] = {67, 98, 23, 11, 47, 13, 94, 58};
49     int i;
50     select_sort (array, ARRAY_SIZE);
51     for (i=0; i < 8; ++i)
52     {
53         printf ("%d ", array[i]);
54     } printf("\n");
55     return (0);
56 }

```

Listing 1.2: *Program to calculate the Gross Pay of a series of workers*

## 1.2.2 Program Output Screenshot

Employee Number	Total Hours	Gross Pay
77621	35	693.00
Employee Number	Total Hours	Gross Pay
82010	16	378.00
Employee Number	Total Hours	Gross Pay
92390	46	986.70
Employee Number	Total Hours	Gross Pay
62396	12	384.00
Employee Number	Total Hours	Gross Pay
89320	9	247.50
Employee Number	Total Hours	Gross Pay
19089	43	756.80
Employee Number	Total Hours	Gross Pay
54209	64	1196.80
Employee Number	Total Hours	Gross Pay
50630	32	704.00
Employee Number	Total Hours	Gross Pay
50630	32	704.00

## 1.3 Problem 3

### 1.3.1 Computer Program

```
1 #include <stdio.h>
2 #define STACK_EMPTY '0'
3 #define STACK_SIZE 20
4
5 void push(char stack[], /* input/output - the stack */
6 char item, /* input - data being pushed onto the stack */
7 int *top, /* input/output - pointer to top of stack */
8 int max_size) /* input - maximum size of stack */
9 {
10     if (*top < max_size-1)
11     {
12         ++(*top);
13         stack[*top] = item;
14     }
15 }
```

```

16 char pop (char stack[], /* input/output - the stack */
17 int *top) /* input/output - pointer to top of stack */
18 {
19     char item; /* value popped off the stack */
20     if (*top >= 0)
21     {
22         item = stack[*top];
23         --(*top);
24     }
25     else
26     {
27         item = STACK_EMPTY;
28     }
29     return (item);
30 }
31 void main(void)
32 {
33     char s [STACK_SIZE];
34     int s_top = -1; // stack is empty
35     int *ptr = &s_top;
36     push(s, 'A', ptr, STACK_SIZE);
37     push(s, 'B', ptr, STACK_SIZE);
38     push(s, 'C', ptr, STACK_SIZE);
39
40     for (int i = 0; i < *ptr; i++)
41     {
42         printf("%c", s[i]);
43     }
44     printf("\n");
45     pop(s, ptr);
46     for (int i = 0; i < *ptr; i++)
47     {
48         printf("%c", s[i]);
49     }
50 }
51
52

```

Listing 1.3: *Program to Calculate the Temperature-Pressure relation for some Temperature and Pressure*

### 1.3.2 Program Output Screenshot

```
Maximum temperature the cylinder can withstand before bursting is 3000.000000
Temperature (K)    Pressure (atm)
-----
300.00            50.00
400.00            66.67
500.00            83.33
600.00            100.00
700.00            116.67
800.00            133.33
900.00            150.00
1000.00           166.67
1100.00           183.33
1200.00           200.00
1300.00           216.67
1400.00           233.33
1500.00           250.00
1600.00           266.67
1700.00           283.33
1800.00           300.00
1900.00           316.67
2000.00           333.33
2100.00           350.00
2200.00           366.67
2300.00           383.33
2400.00           400.00
2500.00           416.67
2600.00           433.33
2700.00           450.00
2800.00           466.67
2900.00           483.33
```