

# Programming Basics (HW#2)

Data Structure

Im Y. Jung



# Problem

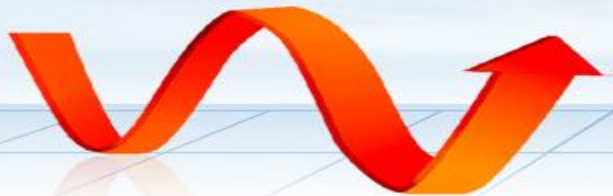
Get 2 sets of 10 integer numbers sorted in ascending order each,  
Print out the 20 numbers sorted in descending order.

- Same numbers should be printed only once.
- You should check whether the input numbers are sorted in ascending order and the number of integers is  $2 * 10$  in total.

If not, you must print out the following error message and terminate your program,

"The input numbers are not in ascending order."

Or, "You must input 2 sets of 10 numbers."



# Problem

## - Execution

Input :

-1, 2, 6, 8, 19, 100, 120, 210, 211, 212

1, 3, 4, 9, 30, 50, 111, 211, 213, 215

Output :

215, 213, 212, 211, 210, 120, 111, 100, 50, 30, 19, 9, 8, 6, 4, 3, 2, 1, -1

Input :

1, 2, 8, 6, 19, 100, 120, 210, 211, 212

Output :

The input numbers are not in ascending order.

Input :

1, 2, 8, 6, 19, 100

Output :

You should input 2 sets of 10 numbers.

## *Example – Problem Analysis (1)*

- Input : 2 \* ( 10 numbers )
- Output : 20 numbers sorted in descending order
- Requirements :
  - **Same numbers should be printed only once.**
  - **Check that the input numbers are sorted in ascending order.**
    - If not, print out an error message.
  - **Check that the number of integers is 2 \* 10 in total.**
    - If not, print out an error message.
- What to do
  - **In/Out Design**
    - Keyboard in/Screen out
  - **Get the data from keyboard and store it, check the inputs**
    - Mixed input of integer and characters (‘,’ and ‘ ‘ )
  - **Sort and print out the ordered numbers at screen**
- What to use
  - **Data/storage Design**
    - `int *in[2]`

## *Example – Problem Analysis (2)*

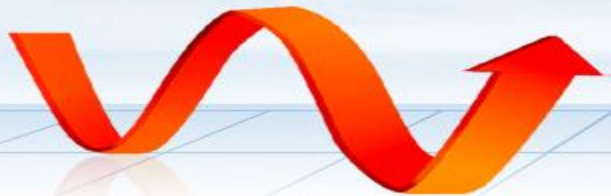
- How to do

- **Program structure**

- Several functions ?
      - Extract and analyze input numbers
      - Sort 20 numbers in descending order
      - Print out the sorted numbers

- **Algorithm**

- Get the input of 10 \* 2 numbers
      - Get the various typed data entered
        - » **scanf, gets, getchar**
    - Extract 10 \* 2 numbers
      - Check the numbers
      - Check the descending order of the inputs
    - Sort and print out 20 numbers
      - Print integers
        - » **printf**





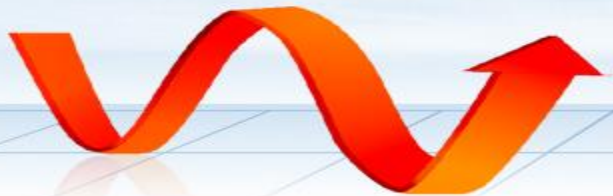
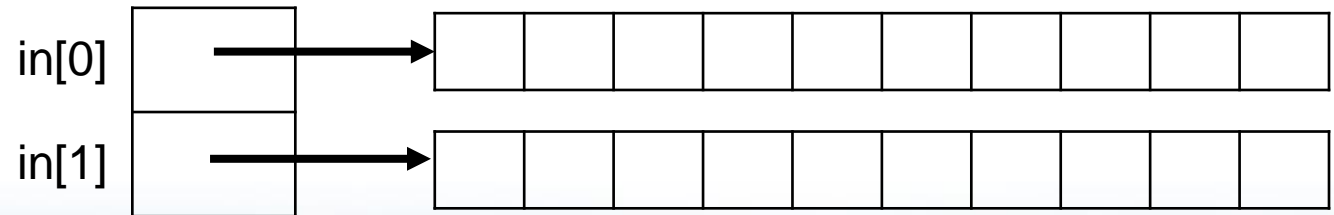
## *Example – Data/storage Design*

**char temp[100];**

**int status;**

**int \*in[2];**

- Get a string which contains 10 numbers mixed with ‘,’ and ‘ ‘
- NORMAL, NO\_ASC, NO\_TOTAL, NO\_MEM
- \*int[2]



## Example – Program Flow (1)

```
char temp[100];
int input[10];
int status = NORMAL;

scanf("%[^\\n]s", temp);

for(token = strtok(temp, d), i=0;
    token != NULL && i < 10 ;
    token = strtok(NULL, d)){

    input[i++] = atoi(token);

    if(i > 1 && input[i-2] > input[i-1]){
        status = NO_ASC;
    }
}

if(i != 10){
    status = NO_TOTAL;
}
```

- Get a string which contains 10 numbers mixed with ‘,’ and ‘ ‘
  - **Extract numbers from the string**
  - **Check that the input numbers are sorted in ascending order**
  - **Check that the number of integers is 2 \* 10 in total**

## Example – Program Flow (2)

```
int output[20];

for(i=MAX-1, j=MAX-1, k=0; i>=0 && j>=0 && k<T_MAX;){
    if( in[0][i] < in[1][j] )
        output[k++] = in[1][j--] ;
    else if(in[0][i] > in[1][j])
        output[k++] = in[0][i--];
    else{
        output[k++] = in[0][i--];
        j--;
    }
}

printf("Output : ");
for(i=0;i<k;i++){
    printf("%d", output[i]);
    if(i+1!=k)
        printf(", ");
}
```

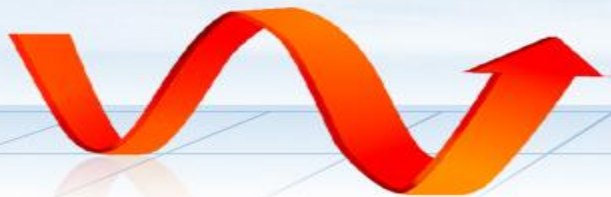
- Sort the numbers in in[0][] and in[1][], and store them in output[]
  - **The same numbers are saved as one number**
- Print out output[]



## Sample, ver. 1 (1/2)

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  #define MAX 10
6  #define T_MAX 20
7
8  #define NORMAL 0
9  #define NO_ASC 1
10 #define NO_TOTAL 2
11 #define NO_MEM 3
12
13 int *getin( int *);
14 void sort( int *in[]);
```

```
16 int main()
17 {
18     int *in[2], status=NORMAL;
19
20     printf("Input 2*10 numbers :#\n");
21     in[0]=getin(&status);
22     if(status==NORMAL)
23         in[1]=getin(&status);
24
25     switch(status)
26     {
27         case NORMAL :
28             sort(in);
29             break;
30         case NO_ASC :    // No Ascending Order
31             printf("The input numbers are not in ascending order!#\n");
32             break;
33         case NO_TOTAL :  // Not 20 numbers
34             printf("You should input 20 numbers in total.#\n");
35             break;
36         case NO_MEM :    // No memory assignment
37             printf("No memory allocation.#\n");
38             break;
39     }
40
41     return 0;
42 }
```



## Sample, ver. 1 (2/2)

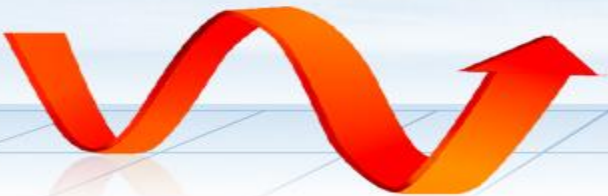
```
44 int *getin(int *status)
45 {
46     char temp[100];
47     int *input = NULL;
48     char d[2] = ",";
49     char *token;
50     int i;
51
52     input = (int *)malloc(sizeof(int)*MAX);
53     if( input != NULL ){
54         scanf("%[^\\n]s", temp);
55
56         for(token = strtok(temp, d), i=0; token != NULL && i < MAX; token = strtok(NULL, d)){
57             input[i++] = atoi(token);
58
59             if(i>1 && input[i-2] > input[i-1]){
60                 *status = NO_ASC;
61                 free(input);
62                 return NULL;
63             }
64         }
65         if(i != MAX){
66             *status = NO_TOTAL;
67             free(input);
68             return NULL;
69         }
70     }
71     else
72         *status = NO_MEM;
73
74     return input;
75 }
```

```
77 void sort(int *in[])
78 {
79     int i, j, k;
80     int output[T_MAX];
81
82     for(i=MAX-1, j=MAX-1, k=0; i>=0 && j>=0 && k<T_MAX;){
83         if( in[0][i] < in[1][j] )
84             output[k++] = in[1][j--];
85         else if(in[0][i] > in[1][j])
86             output[k++] = in[0][i--];
87         else{
88             output[k++] = in[0][i--];
89             j--;
90         }
91     }
92
93     if(i>=0)
94         while(i<MAX && k<T_MAX)
95             output[k++] = in[0][i--];
96
97     if(j>=0)
98         while(j<MAX && k<T_MAX)
99             output[k++] = in[1][j--];
100
101     printf("Output : ");
102     for(i=0; i<k; i++){
103         printf("%d", output[i]);
104         if(i+1!=k)
105             printf(", ");
106     }
107
108     free(in[0]);
109     free(in[1]);
110 }
```

## Sample, ver. 2 (1/2)

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  #define MAX 10
6  #define T_MAX 20
7
8  #define NORMAL 0
9  #define NO_ASC 1
10 #define NO_TOTAL 2
11 #define NO_MEM 3
12
13 int **getin(int *);
14 void sort(int *in[]);
```

```
16 int main()
17 {
18     int **in, status=NORMAL;
19
20     in=getin(&status);
21
22     switch(status)
23     {
24         case NORMAL :
25             sort(in);
26             break;
27         case NO_ASC :    // No Ascending Order
28             printf("The input numbers are not in ascending order!\n");
29             break;
30         case NO_TOTAL :    // Not 20 numbers
31             printf("You should input 20 numbers in total.\n");
32             break;
33         case NO_MEM :    // No memmory allocation
34             printf("No memory allocation.\n");
35             break;
36     }
37
38     return 0;
39 }
```



# Sample, ver. 2 (2/2)

```
41 int **getin(int *status)
42 {
43     char temp[100];
44     int **input = NULL;
45     char d[2] = ",";
46     char *token;
47     int i, line=0;
48
49     input = (int **)malloc(sizeof(int *)*2);
50
51     if( input != NULL ){
52         printf("Input 2*10 numbers :\n");
53
54         while(line<2){
55             if((input[line] = (int *)malloc(sizeof(int)*MAX))==NULL)
56             {
57                 *status = NO_MEM;
58                 return input;
59             }
60             scanf(" %[^\\n]s", temp);
61
62             for(token = strtok(temp, d), i=0; token != NULL && i < MAX; token = strtok(NULL, d)){
63                 input[line][i++] = atoi(token);
64
65                 if(i>1 && input[line][i-2] > input[line][i-1]){
66                     *status = NO_ASC;
67                     free(input[line]);
68                     free(input);
69                     return NULL;
70                 }
71             }
72             if(i != MAX){
73                 *status = NO_TOTAL;
74                 free(input[line]);
75                 free(input);
76                 return NULL;
77             }
78             line++;
79         }
80     }
81     else
82         *status = NO_MEM;
83
84     return input;
85 }
```

```
87 void sort(int *in[])
88 {
89     int i, j, k;
90     int output[T_MAX];
91
92     for(i=MAX-1, j=MAX-1, k=0; i>=0 && j>=0 && k<T_MAX;){
93         if( in[0][i] < in[1][j] )
94             output[k++] = in[1][j--] ;
95         else if(in[0][i] > in[1][j])
96             output[k++] = in[0][i--];
97         else{
98             output[k++] = in[0][i--];
99             j--;
100         }
101     }
102
103     if(i>=0)
104         while(i<MAX && k<T_MAX)
105             output[k++] = in[0][i--];
106
107     if(j>=0)
108         while(j<MAX && k<T_MAX)
109             output[k++] = in[1][j--];
110
111     printf("Output : ");
112     for(i=0; i<k; i++){
113         printf("%d", output[i]);
114         if(i+1!=k)
115             printf(", ");
116     }
117
118     free(in[0]);
119     free(in[1]);
120     free(in);
121 }
```