# **Programming Basics (HW#2)**

**Data Structure** 



### **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]

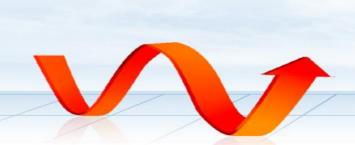


### Example – Problem Analysis (2)

- 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



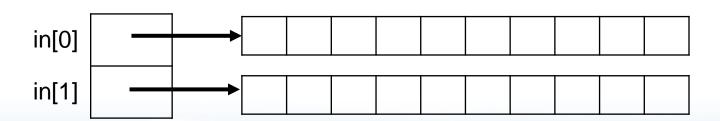
## Example – Data/storage Design

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)

16

18

19

20

24

25

26

29

30

33

34 35

36 37

38

39 40

```
⊟#include <stdio.h>
       #include <string.h>
       #include <stdlib.h>
       #define MAX 10
       #define T_MAX 20
       #define NORMAL O
       #define NOLASC 1
       #define NO_TOTAL 2
10
       #define NO_MEM 3
1.1
12
13.
        int *getin(int *);
       void sort(int *in[]);
14
```

```
⊡int main()
     int *in[2], status=NORMAL;
     printf("Input 2*10 numbers :\");
     in[0]=getin(&status);
     if(status==NORMAL)
          in[1]=getin(&status);
     switch(status)
         case NORMAL :
             sort(in);
             break:
         case NO_ASC : // No Ascending Order
              printf("The input numbers are not in ascending order!\"n");
             break:
         case NOLTOTAL : // Not 20 numbers
             printf("You should input 20 numbers in total.\"n");
             break:
         case NOLMEM :
                            // No memory assignment
             printf("No memory allocation.\"n");
             break:
     return 0;
```



# Sample, ver. 1 (2/2)

```
*getin(int *status)
45 - -
        char temp[100];
        int *input = NULL;
        char d[2] =",";
        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)){</pre>
57
                input[i++]=atoi(token);
58
59 -
                if(i>1 && input[i-2] > input[i-1]){
                     *status = NO ASC;
60
61
                    free(input);
62
                    return NULL;
63
64
65 -
            if(i != MAX){
                 *status = NO_TOTAL;
                free(input);
68
                return NULL;
69
             *status = NO MEM;
        return input;
```

```
void sort(int *in[])
 78 - {
         int i, j, k;
         int output[T_MAX];
81
         for(i=MAX-1, j=MAX-1, k=0; i>=0 && <math>j>=0 && k<T MAX;){
82 -
             if( in[0][i] < in[1][j] )</pre>
83
                  output[k++] = in[1][j--];
84
              else if(in[0][i] > in[1][j])
                  output[k++] = in[0][i--];
87 -
             else{
                  output[k++] = in[0][i--];
                  j--;
90
91
92
93
         if(i>=0)
             while(i<MAX && k<T MAX)
94
                  output[k++] = in[0][i--];
95
96
         if(j>=0)
             while(j<MAX && k<T MAX)</pre>
98
                  output[k++] = in[1][j--];
100
101
         printf("Output : ");
102 -
         for(i=0;i<k;i++){</pre>
103
             printf("%d", output[i]);
104
             if(i+1!=k)
105
                 printf(", ");
106
107
108
         free(in[0]);
109
         free(in[1]);
110
                                                         10
```

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

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

```
⊟int main()
16
17
18
            int **in. status=NORMAL)
19
            in=getin(&status);
20.
           switch(status)
22
24
                case NORMAL :
                    sort(in);
26
                    break:
                                  // No Ascending Order
                case NOLASC :
                    printf("The input numbers are not in ascending order!\"n");
28
29
                    break:
                case NO IOIAL : // Not 20 numbers
30.
                    printf("You should input 20 numbers in total.\"n");
31
                    break:
33
                case NO_MEM : // No memmory allocation
34
                    printf("No memory allocation.\"n");
35
                    break:
36
            return 00
38
39
```

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

```
int **getin(int *status)
42 -
43
        char temp[100];
        int **input = NULL;
45
        char d[2] =",";
        char *token;
        int i, line=0;
48
49
        input = (int **)malloc(sizeof(int *)*2);
50
51 -
        if( input != NULL ){
             printf("Input 2*10 numbers :\n");
54
55
56
57
             while(line<2){</pre>
                 if((input[line] = (int *)malloc(sizeof(int)*MAX))==NULL)
                      *status = NO MEM;
58
59
60
61
62
63
64
65
66
67
                      return input;
                 scanf(" %[^\n]s", temp);
                 for(token = strtok(temp, d), i=0; token != NULL && i < MAX; token = strtok(NULL, d)){</pre>
                      input[line][i++]=atoi(token);
                      if(i>1 && input[line][i-2] > input[line][i-1]){
                           *status = NO_ASC;
                          free(input[line]);
68
69
                          free(input);
                          return NULL;
                 if(i != MAX){
                      *status = NO_TOTAL;
                      free(input[line]);
                      free(input);
                      return NULL;
                 line++;
              *status = NO MEM;
84
        return input;
```

```
void sort(int *in[])
         int i, j, k;
         int output[T_MAX];
         for(i=MAX-1, j=MAX-1, k=0; i>=0 && <math>j>=0 && k<T_MAX;){
              if( in[0][i] < in[1][j] )</pre>
                  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--;
         if(i>=0)
              while(i<MAX && k<T MAX)</pre>
                  output[k++] = in[0][i--];
         if(j>=0)
             while(j<MAX && k<T MAX)</pre>
                  output[k++] = in[1][j--];
110
111
         printf("Output : ");
112
         for(i=0;i<k;i++){
113
                   f("%d", output[i]);
114
              if(i+1!=k)
115
                  printf(", ");
116
117
118
         free(in[0]);
119
         free(in[1]);
120
             e(in);
121 }
```