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**

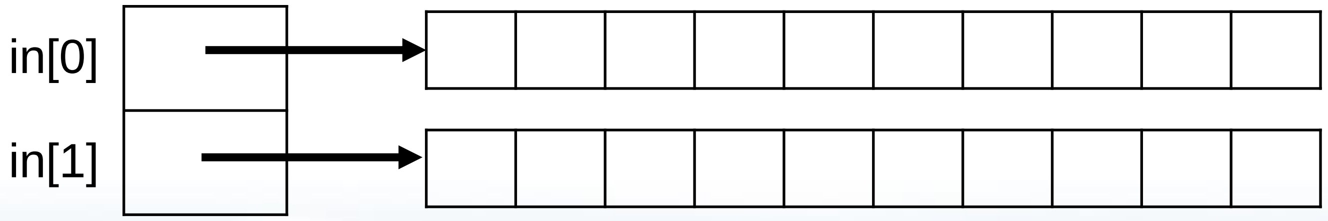
Example – Data/storage Design

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;
}
```

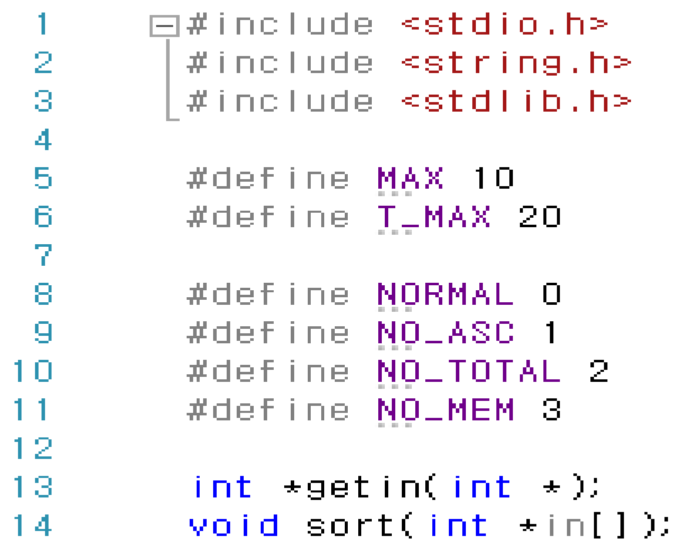
- Get a string which contains 10 numbers mixed with ‘,’ and ‘ ‘
  - **Extract numbers from the string**
  - **Check that the input numbers are sorted in ascending order**
  - **Check that the number of integers is 2 * 10 in total**

Example – Program Flow (2)

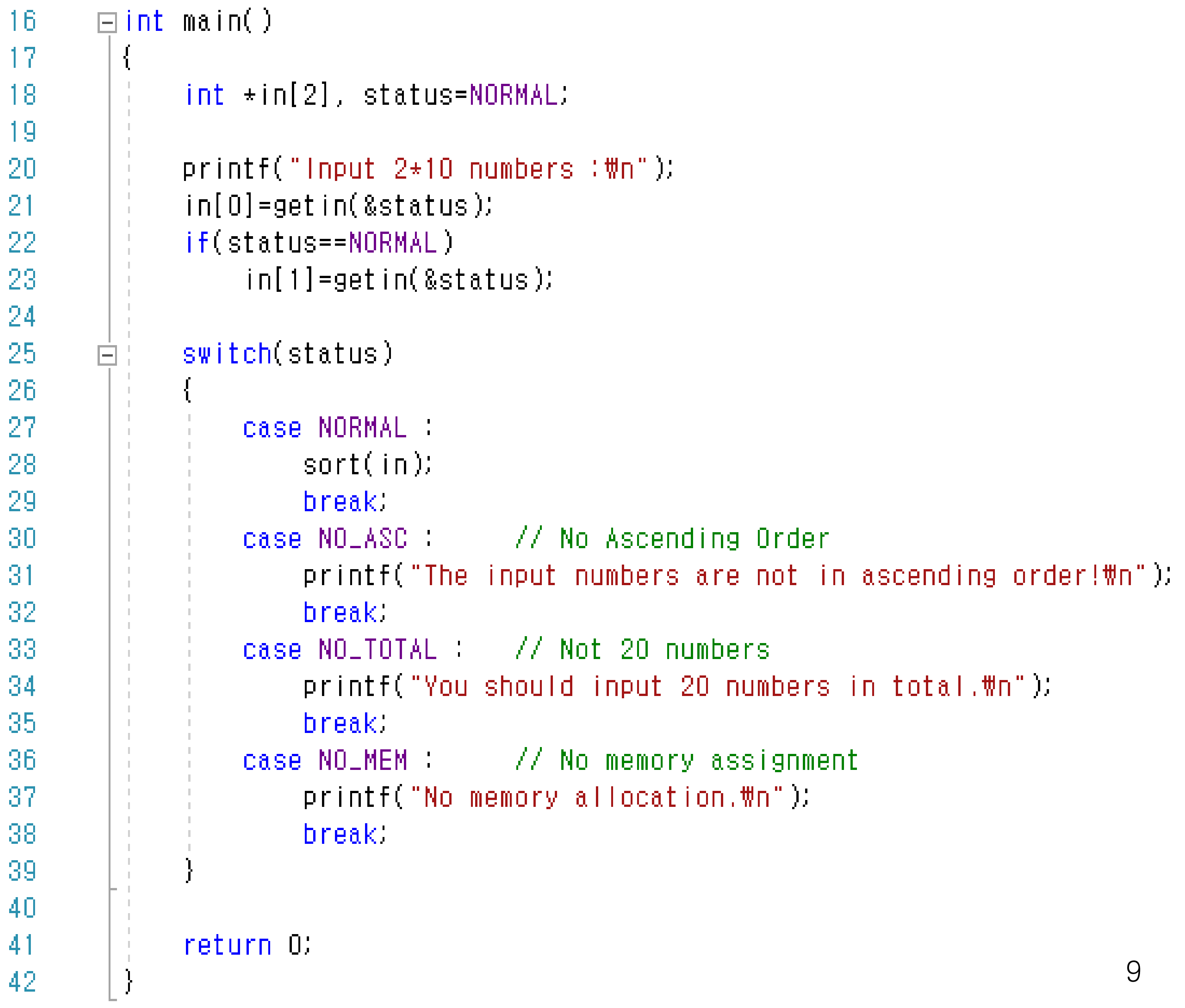
```
int output[20];

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)

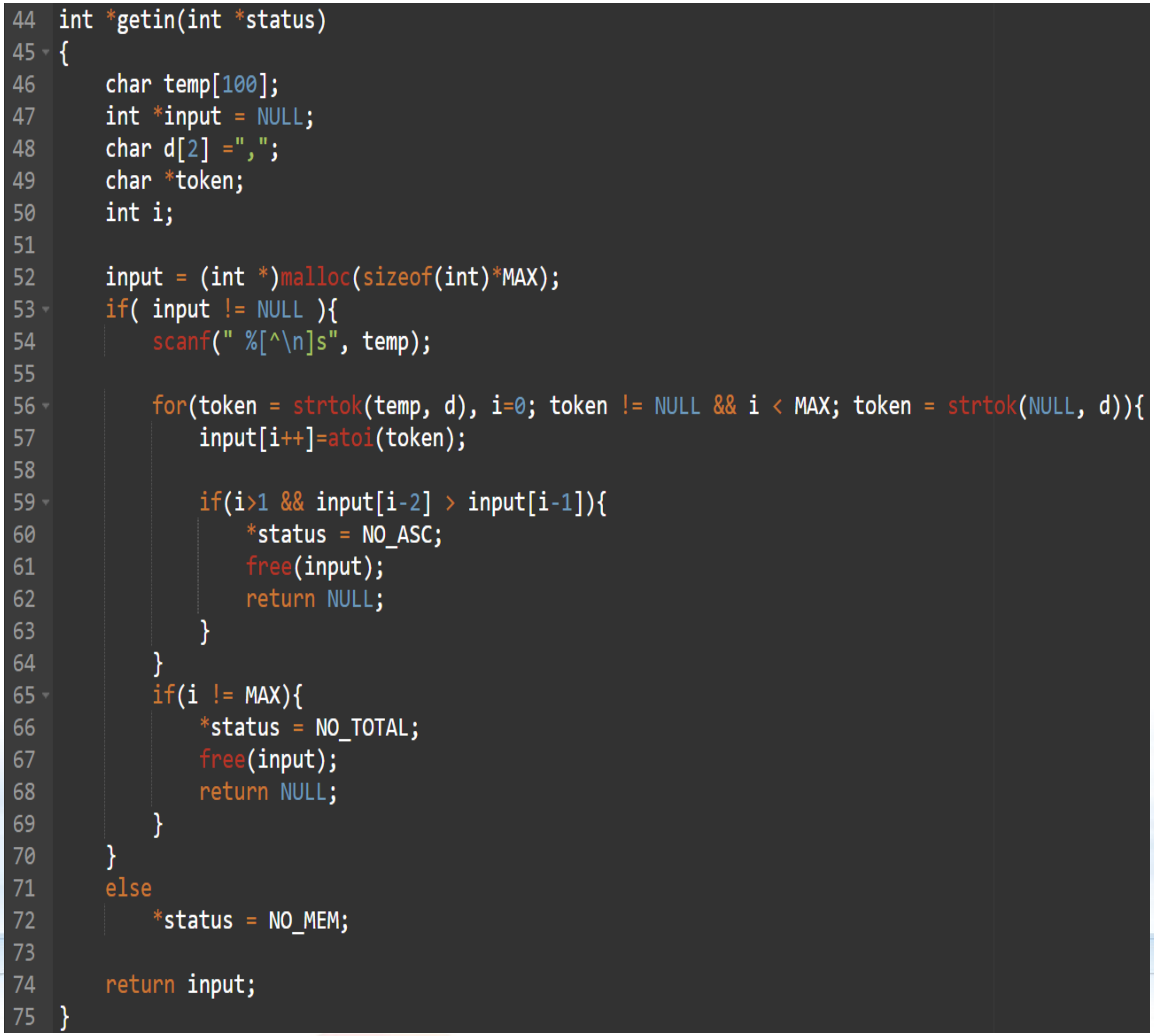


```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX 10
#define T_MAX 20

#define NORMAL 0
#define NO_ASC 1
#define NO_TOTAL 2
#define NO_MEM 3

int *getin(int *);
void sort(int *in[]);
```

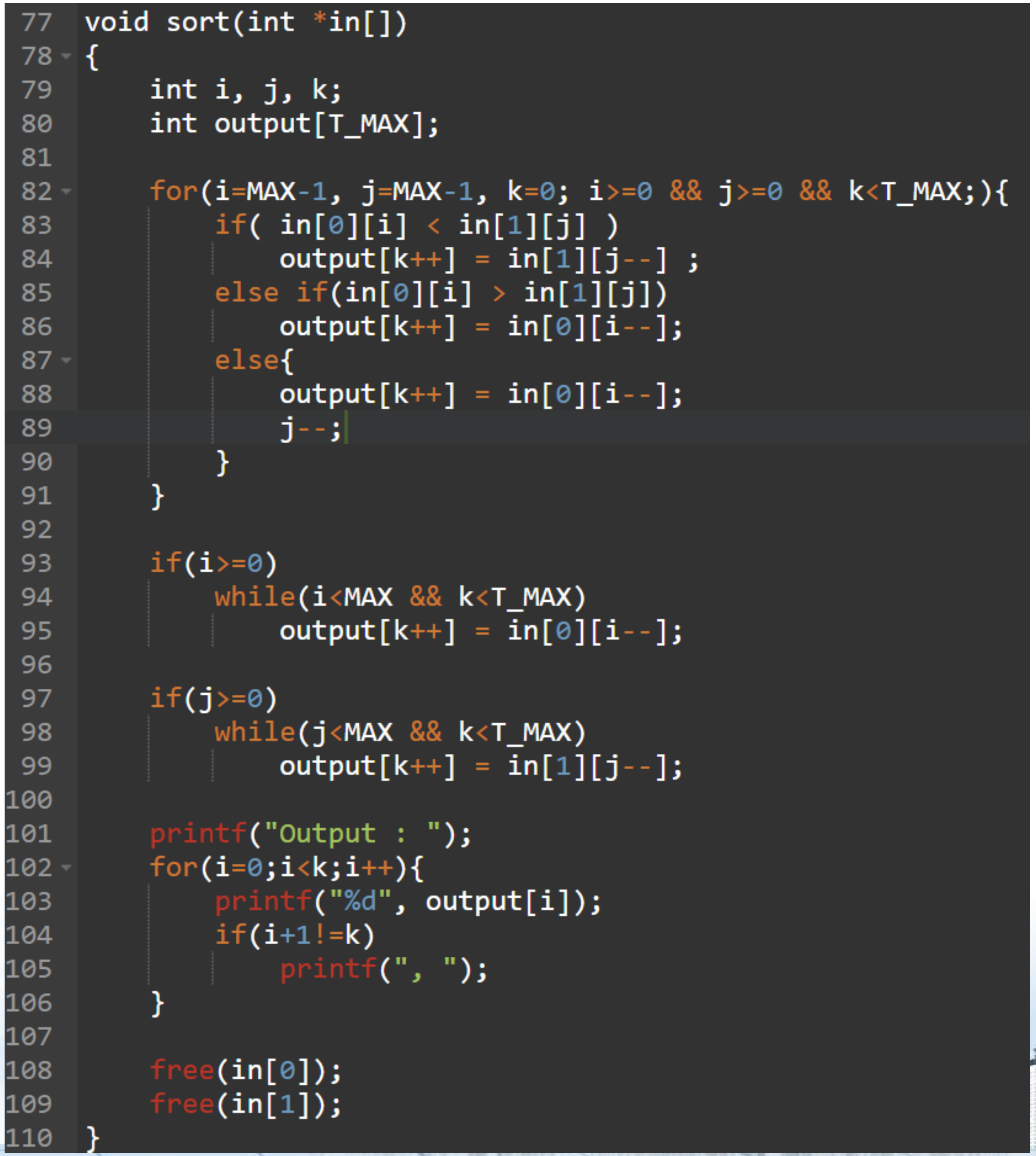


```c
int main()
{
    int *in[2], status=NORMAL;

    printf("Input 2*10 numbers :₩n");
    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 NO_TOTAL :     // Not 20 numbers
            printf("You should input 20 numbers in total.₩n");
            break;
        case NO_MEM :       // No memory assignment
            printf("No memory allocation.₩n");
            break;
    }

    return 0;
}
```

Sample, ver. 1 (2/2)

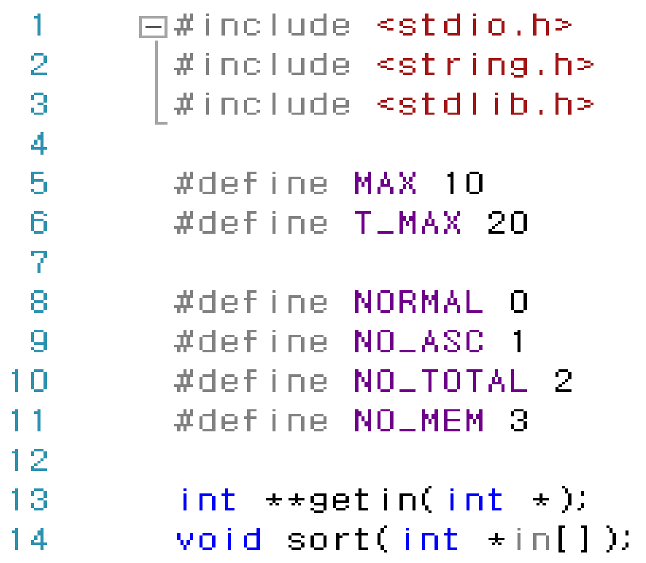


```c
int *getin(int *status)
{
    char temp[100];
    int *input = NULL;
    char d[2] =",";
    char *token;
    int i;

    input = (int *)malloc(sizeof(int)*MAX);
    if( input != NULL ){
        scanf(" %[^\n]s", temp);

        for(token = strtok(temp, d), i=0; token != NULL && i < MAX; token = strtok(NULL, d)){
            input[i++]=atoi(token);

            if(i>1 && input[i-2] > input[i-1]){
                *status = NO_ASC;
                free(input);
                return NULL;
            }
        }
        if(i != MAX){
            *status = NO_TOTAL;
            free(input);
            return NULL;
        }
    }
    else
        *status = NO_MEM;

    return input;
}
```

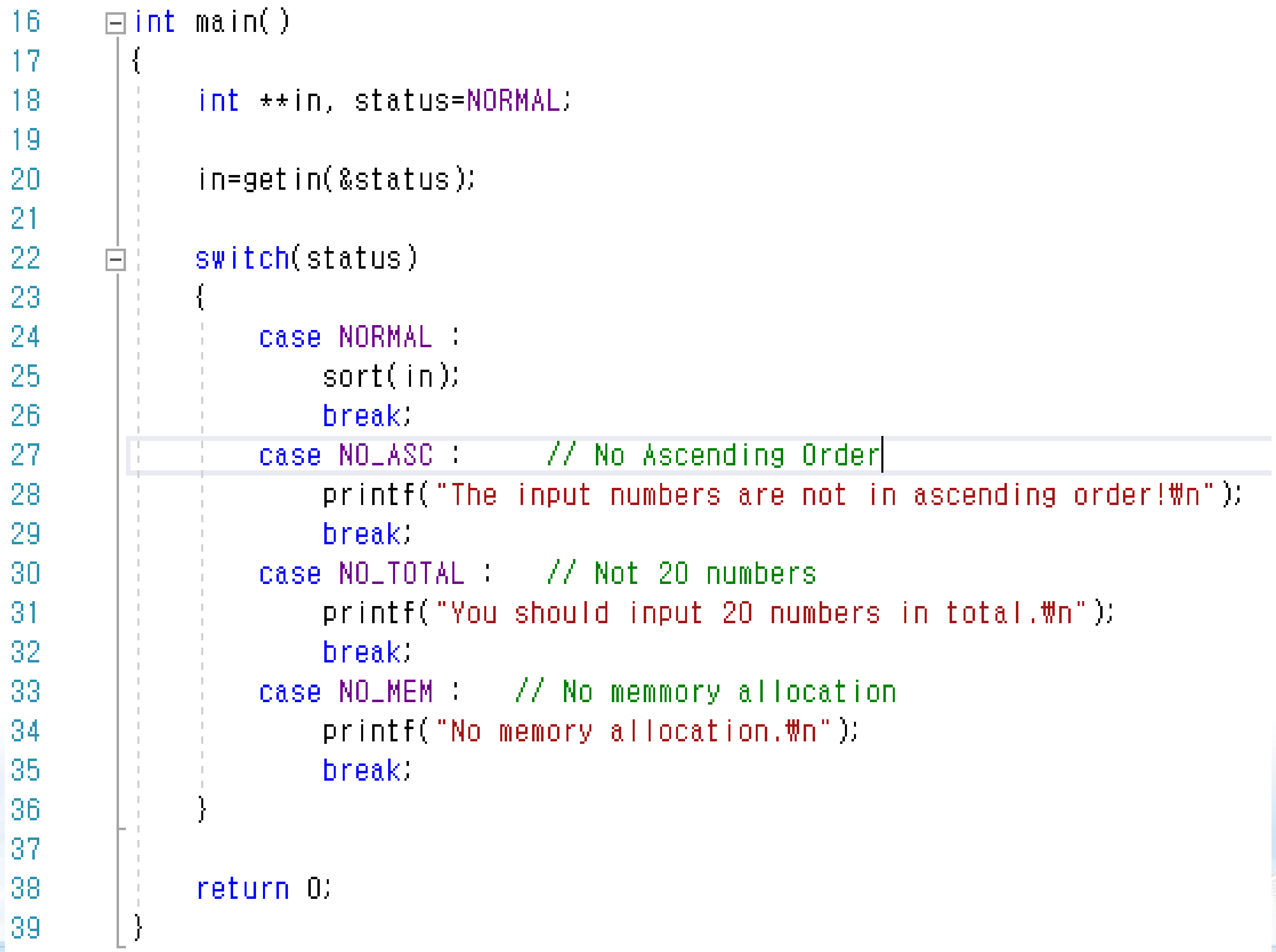


```c
void sort(int *in[])
{
    int i, j, k;
    int output[T_MAX];

    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--;
        }
    }

    if(i>=0)
        while(i<MAX && k<T_MAX)
            output[k++] = in[0][i--];

    if(j>=0)
        while(j<MAX && k<T_MAX)
            output[k++] = in[1][j--];

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

    free(in[0]);
    free(in[1]);
}
```

Sample, ver. 2 (1/2)

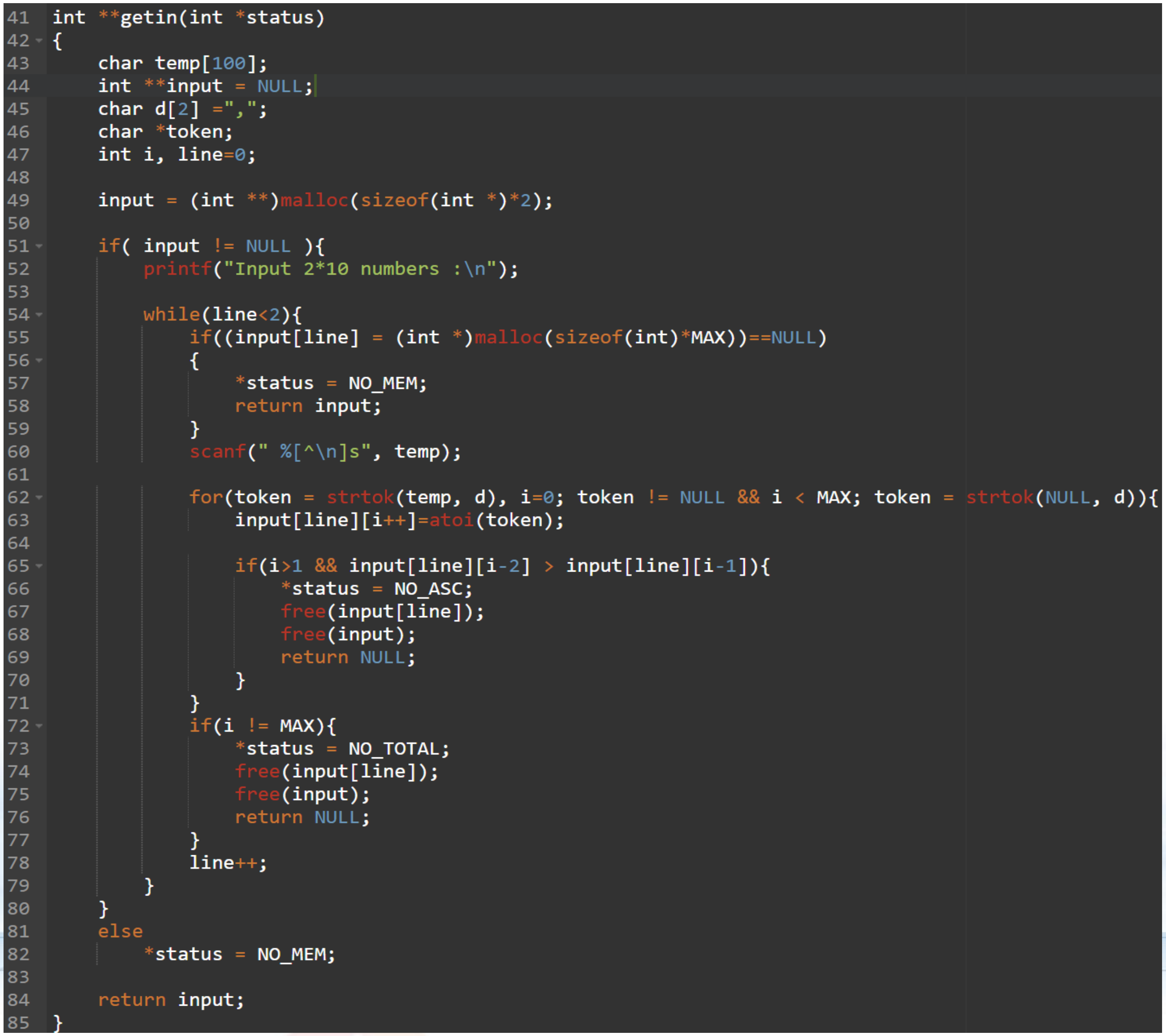


```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX 10
#define T_MAX 20

#define NORMAL 0
#define NO_ASC 1
#define NO_TOTAL 2
#define NO_MEM 3

int **getin(int *);
void sort(int *in[]);
```

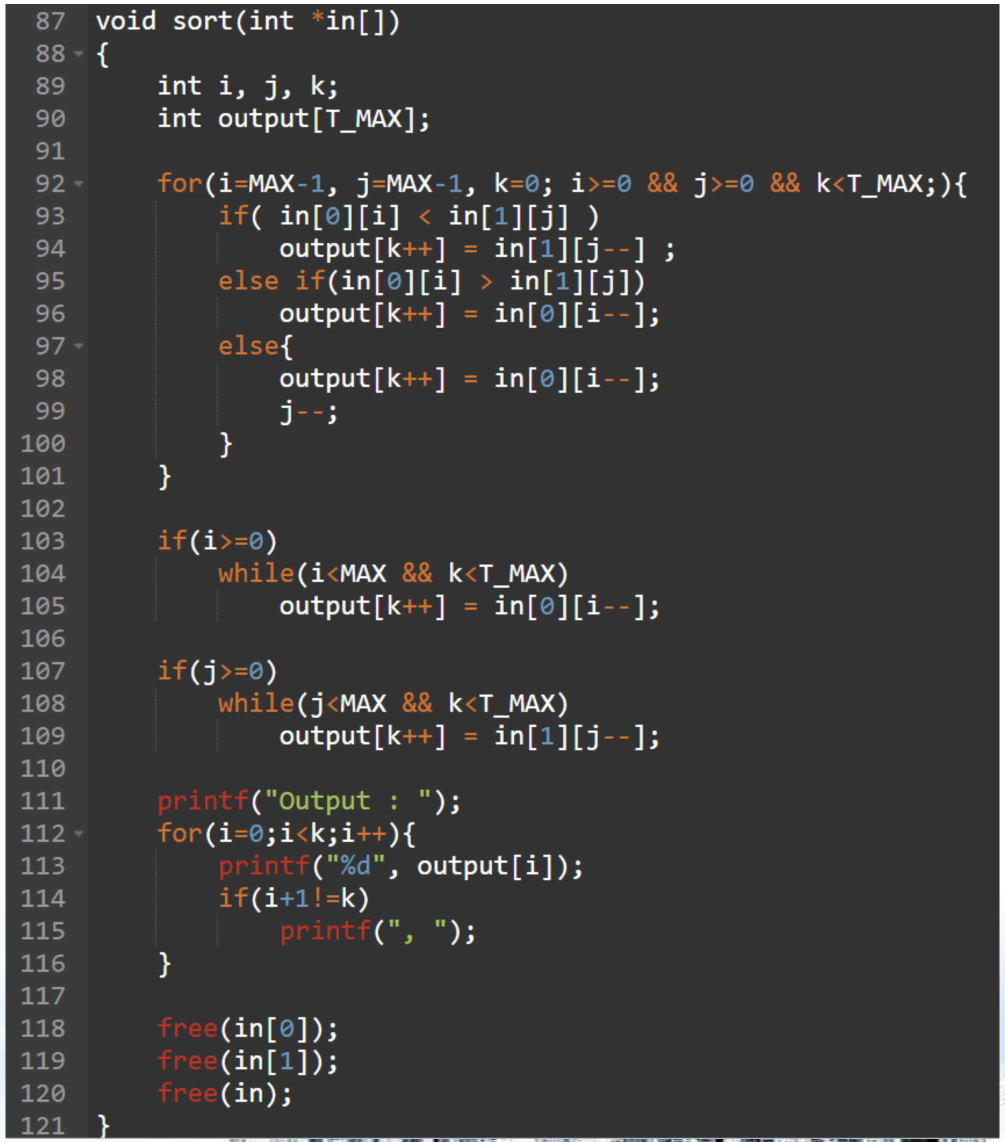


```c
int main()
{
    int **in, status=NORMAL;

    in=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 NO_TOTAL :    // Not 20 numbers
            printf("You should input 20 numbers in total.₩n");
            break;
        case NO_MEM :    // No memmory allocation
            printf("No memory allocation.₩n");
            break;
    }

    return 0;
}
```

Sample, ver. 2 (2/2)



```c
int **getin(int *status)
{
    char temp[100];
    int **input = NULL;
    char d[2] =",";
    char *token;
    int i, line=0;

    input = (int **)malloc(sizeof(int *)*2);

    if( input != NULL ){
        printf("Input 2*10 numbers :\n");

        while(line<2){
            if((input[line] = (int *)malloc(sizeof(int)*MAX))==NULL)
            {
                *status = NO_MEM;
                return input;
            }
            scanf(" %[^\n]s", temp);

            for(token = strtok(temp, d), i=0; token != NULL && i < MAX; token = strtok(NULL, d)){
                input[line][i++]=atoi(token);

                if(i>1 && input[line][i-2] > input[line][i-1]){
                    *status = NO_ASC;
                    free(input[line]);
                    free(input);
                    return NULL;
                }
            }
            if(i != MAX){
                *status = NO_TOTAL;
                free(input[line]);
                free(input);
                return NULL;
            }
            line++;
        }
    }
    else
        *status = NO_MEM;

    return input;
}
```



```c
void sort(int *in[])
{
    int i, j, k;
    int output[T_MAX];

    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--;
        }
    }

    if(i>=0)
        while(i<MAX && k<T_MAX)
            output[k++] = in[0][i--];

    if(j>=0)
        while(j<MAX && k<T_MAX)
            output[k++] = in[1][j--];

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

    free(in[0]);
    free(in[1]);
    free(in);
}
```