Programming Practice

2018-10-25

Week 8

Practice Lecture

Integer Overflow

- When producing the 'sum', 'total amount', 'a big number', etc, calculate the **maximum possible upper bound** first.

- If it is greater than (about) $\mathbf{2 * 10^9}$ => IntegerOverflow will occur.

Integer Overflow

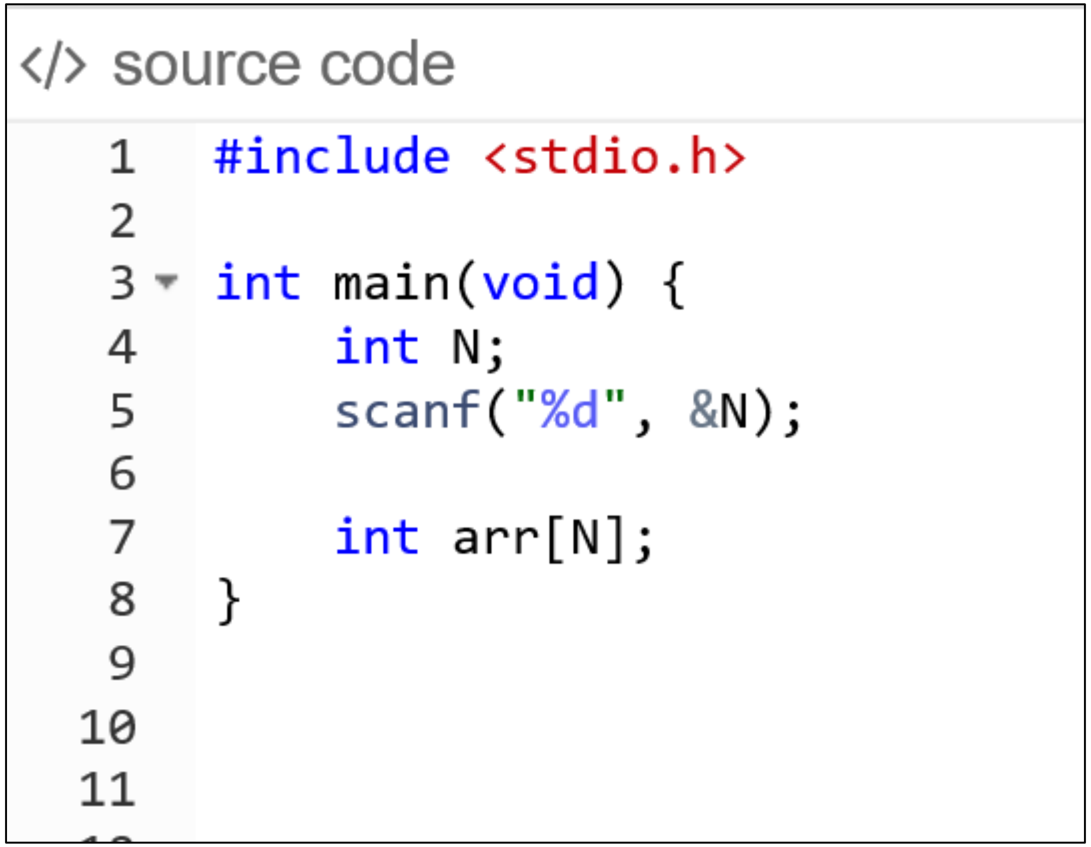
- IntegerOverflow is **not** an error. (The code will compile normally.)
- You must prevent it **yourself**.

➔ Use `long long` type variable (for the sum/total/etc.)

Print with the format `"%lld"` within `printf()` function.

Array declaration

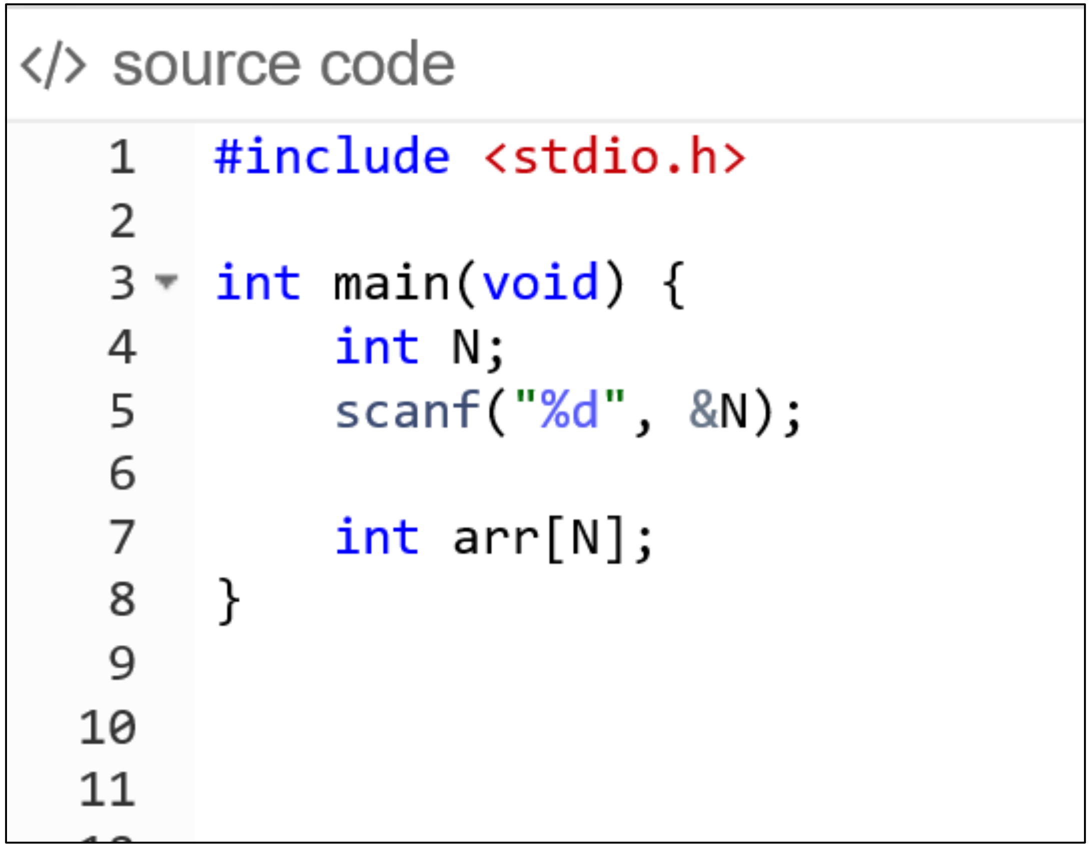
- This is **not** the correct way:



```c
#include <stdio.h>

int main(void) {
    int N;
    scanf("%d", &N);

    int arr[N];
}
```

Array declaration

- This is **not** the correct way:



```c
#include <stdio.h>

int main(void) {
    int N;
    scanf("%d", &N);

    int arr[N];
}
```

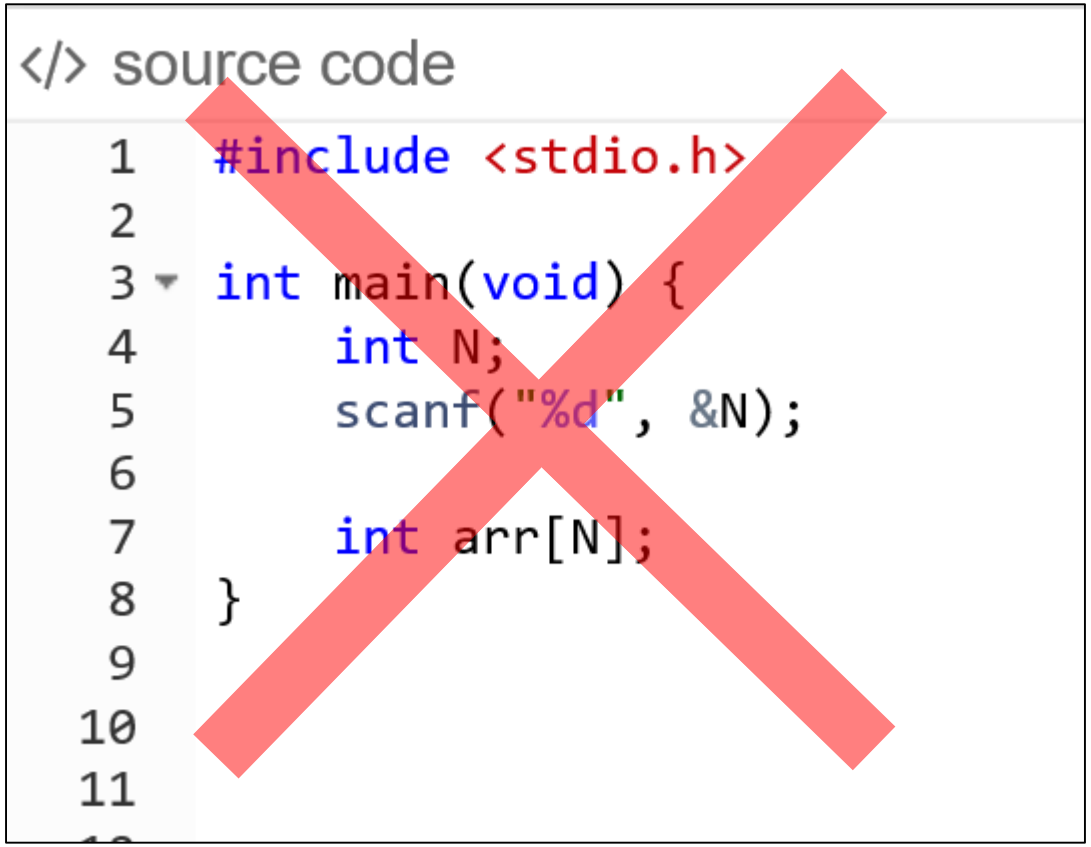
Why?

- `scanf()` is done at ***run-time***

but

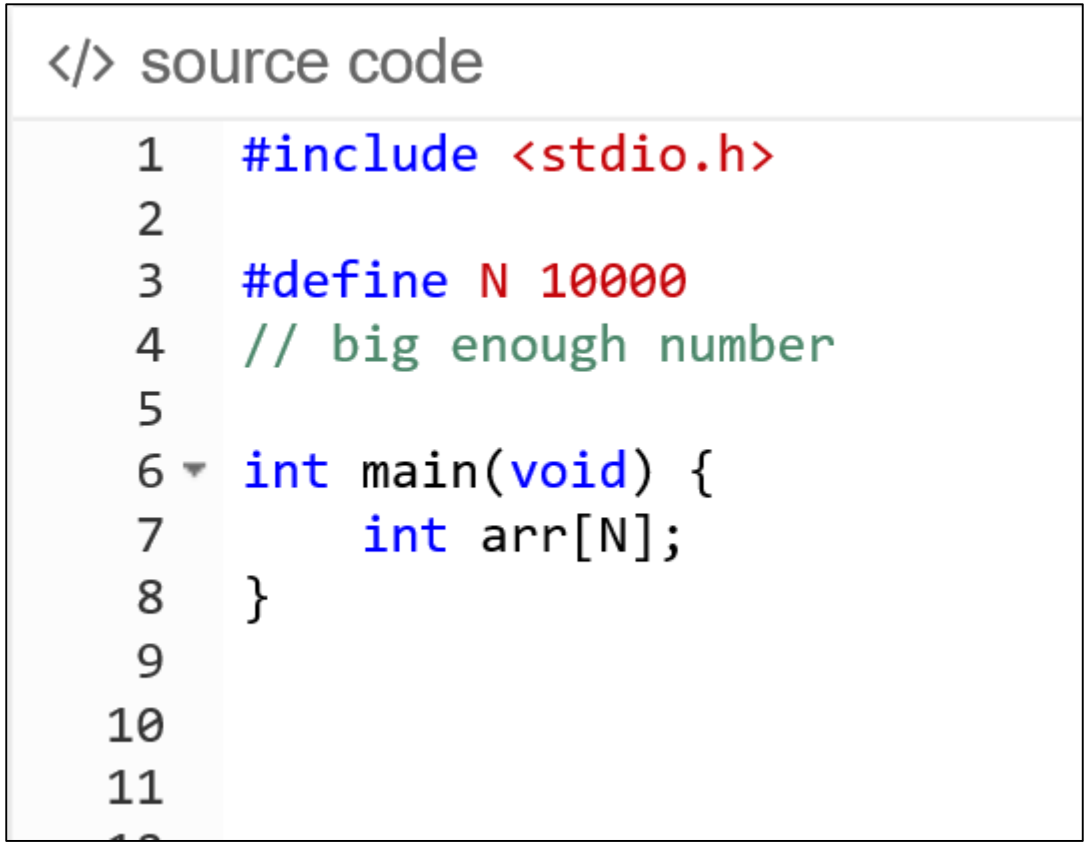
- Array memory allocation
  is done at ***compile-time***

Array declaration



```
#include <stdio.h>

int main(void) {
    int N;
    scanf("%d", &N);

    int arr[N];
}
```

The standard way:



```
#include <stdio.h>

#define N 10000
// big enough number

int main(void) {
    int arr[N];
}
```

Array declaration

- But.. *some* c compilers have *changed* to allow both.

- Depends on the compiler you're using.
  (May be best to do it the standard way.)

Array declaration

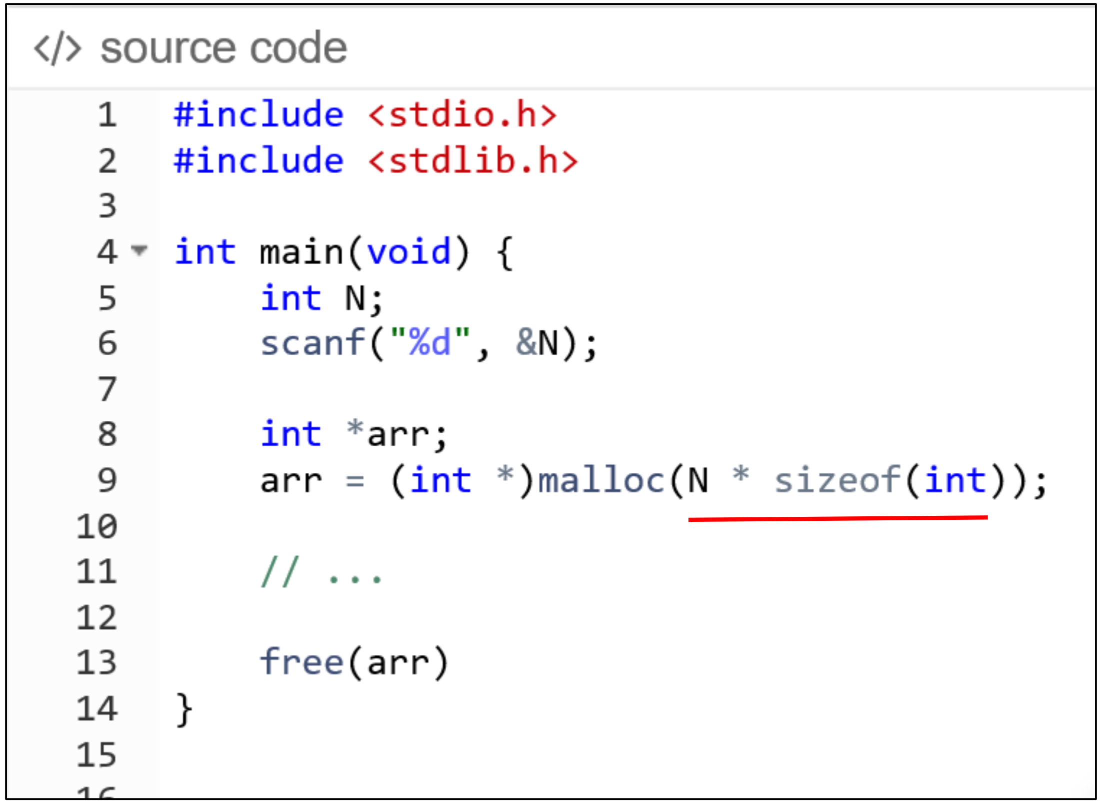
- **OR** use `malloc/calloc` (dynamic memory allocation)

  → Memory is allocated during ***run-time***.

- Makes it possible to declare arrays with non-constant lengths.

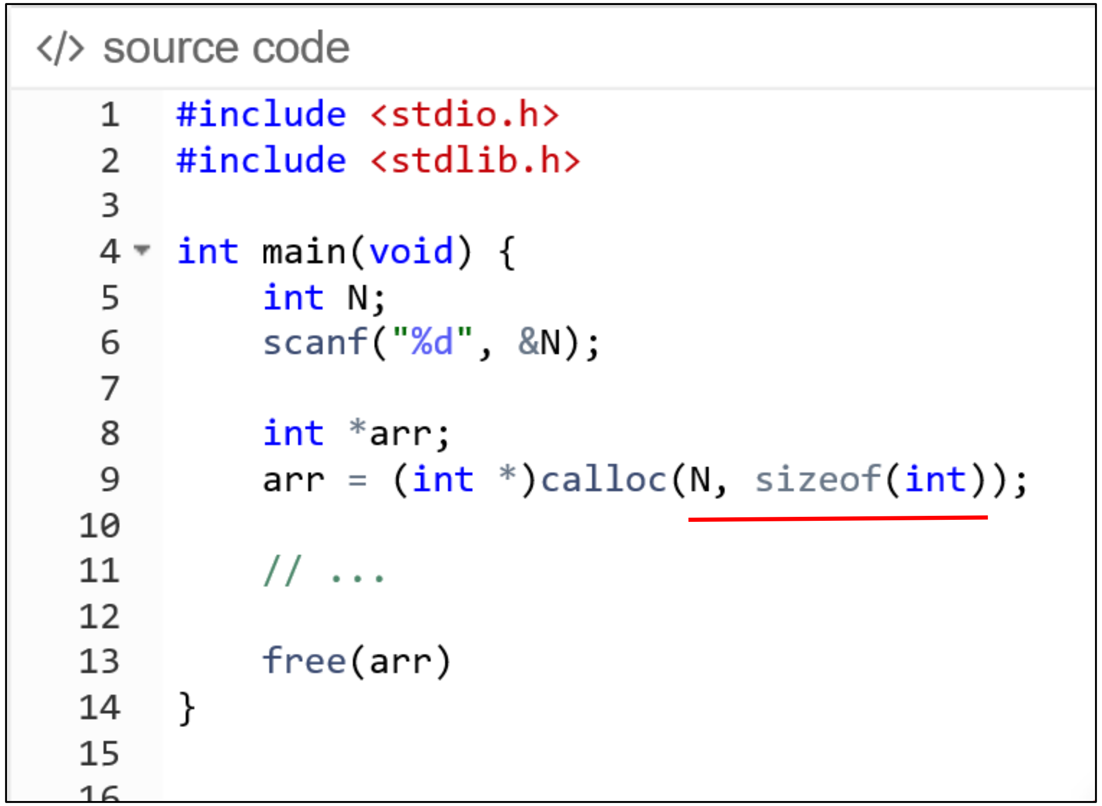
Array declaration

malloc(total_size):



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

int main(void) {
    int N;
    scanf("%d", &N);

    int *arr;
    arr = (int *)malloc(N * sizeof(int));

    // ...

    free(arr)
}

```

calloc(num, size):



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

int main(void) {
    int N;
    scanf("%d", &N);

    int *arr;
    arr = (int *)calloc(N, sizeof(int));

    // ...

    free(arr)
}

```

Array declaration

`malloc:`

Function prototype:

```
void *malloc(size_t size);
```

- Does not initialize elements.

`calloc:`

Function prototype:

```
void *calloc(size_t num, size_t size);
```

- Initializes all elements to 0.

Homework Problems

1. Substring Counter
2. Matrix Multiplication
3. Merging Two Arrays
4. Corpus

Problem. **1**

Substring Counter

Description

String Y is **substring** of string X if and only if X contains Y.
For example, “abc”, “bcb”, “cba” are substrings of “abcba”;
but “aca” is not a substring of “abcba”.

The same substring can appear several times, and your task is to count them. For example, “a” appears three times in “aaa”, while “aa” appears two times in “aaa”.

Given string A and B, count how many times B appears in A.
A and B can contain spaces.

Input

First line contains a string A. Second line contains a string B.
It is guaranteed that $1 \leq (\text{length B}) \leq (\text{length A}) \leq 3000$.

Output

Print a single integer which denotes how many times B appears in A.

Sample

```
[input]
banana
ana
[output]
2
```

```
[input]
aaa aaa
aa
[output]
4
```

Problem. **2**

Matrix Multiplication

Description

Given two matrices A and B, calculate AB (which denotes the multiplication of the matrices A and B).

Size of A is N × M and size of B is M × K.
N, M, K will be given as an input.

Every element of all matrices will be
in the range of [-1000, 1000].

Input

First line contains three integers: N, M, K ($1 \leq N, M, K \leq 200$).
The following N lines contain M integers each, the elements of A.
Following M lines contain K integers each, the elements of B.

Output

Print the result of AB, one row per line. Separate elements of the same row with a single space in between.

Sample

```
[input]
2 3 4
9 8 7
6 5 4
10 11 12 13
14 15 16 17
18 19 20 21
[output]
328 352 376 400
202 217 232 247
```

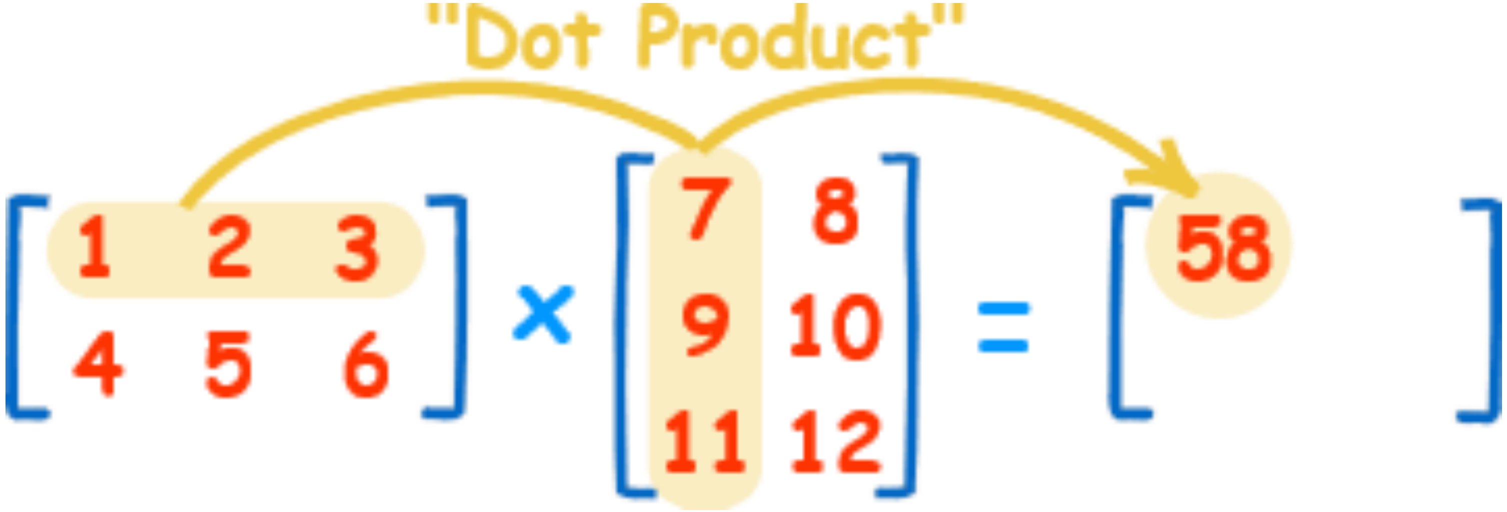
Matrix multiplication

- What is a matrix?

$$\underset{3x4}{A} = \begin{bmatrix} 21 & 62 & 33 & 93 \\ 44 & 95 & 66 & 13 \\ 77 & 38 & 79 & 33 \end{bmatrix}$$

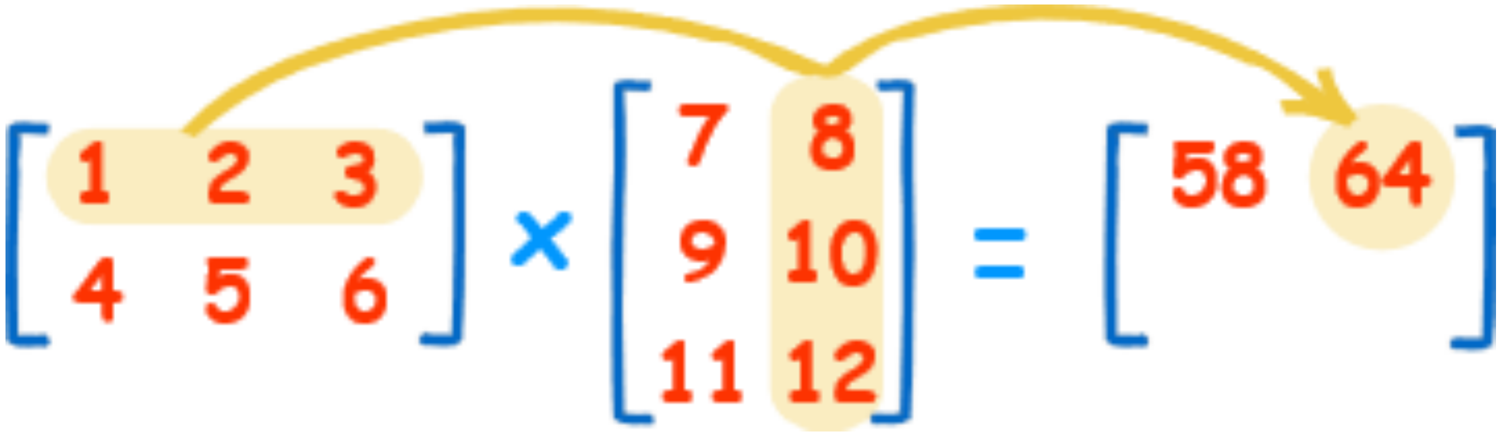
Matrix multiplication

- How is 'matrix multiplication' defined?



Matrix multiplication

- How is 'matrix multiplication' defined?



Matrix multiplication

- How is 'matrix multiplication' defined?

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix} \checkmark$$

Matrix multiplication

- General formula

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} u & v \\ w & x \\ y & z \end{bmatrix}$$

A(2 × 3 matrix) ∗ B(3 × 2 matrix)

$$= \begin{bmatrix} au + bw + cy & av + bx + cz \\ du + ew + fy & dv + ex + fz \end{bmatrix}$$

= C(2 × 2 matrix)

Matrix multiplication

- General formula

Must be equal

$$(x \times y) \text{ matrix} \quad * \quad (y \times z) \text{ matrix}$$

$$= \ (x \times z) \text{ matrix}$$

Matrix multiplication

- General formula

$$\begin{aligned} A &: x \times y \\ B &: y \times z \\ C &: x \times z \end{aligned}$$

$$C_{ij} = \sum_{k=1}^{y} (A_{ik} \times B_{kj})$$

Problem. **3**

Merging Two Arrays

Description

A partially sorted array will be given. It consists of two parts, and each part is already sorted on its own. Make the partially sorted array into a fully sorted array, by merging the two parts.

Length of the entire array N, and length of the first part M, will be given as input. (M ≤ N-1)
(Note that M also == the *starting index* of the second part.)

You have to implement `merge()` function in your merge.c file.
Do not include `main()` function in your submission code.

Input

First line contains two integers N, M. (1 ≤ M < N ≤ 100000)
Second line contains N integers.

Output

Print the fully sorted array.

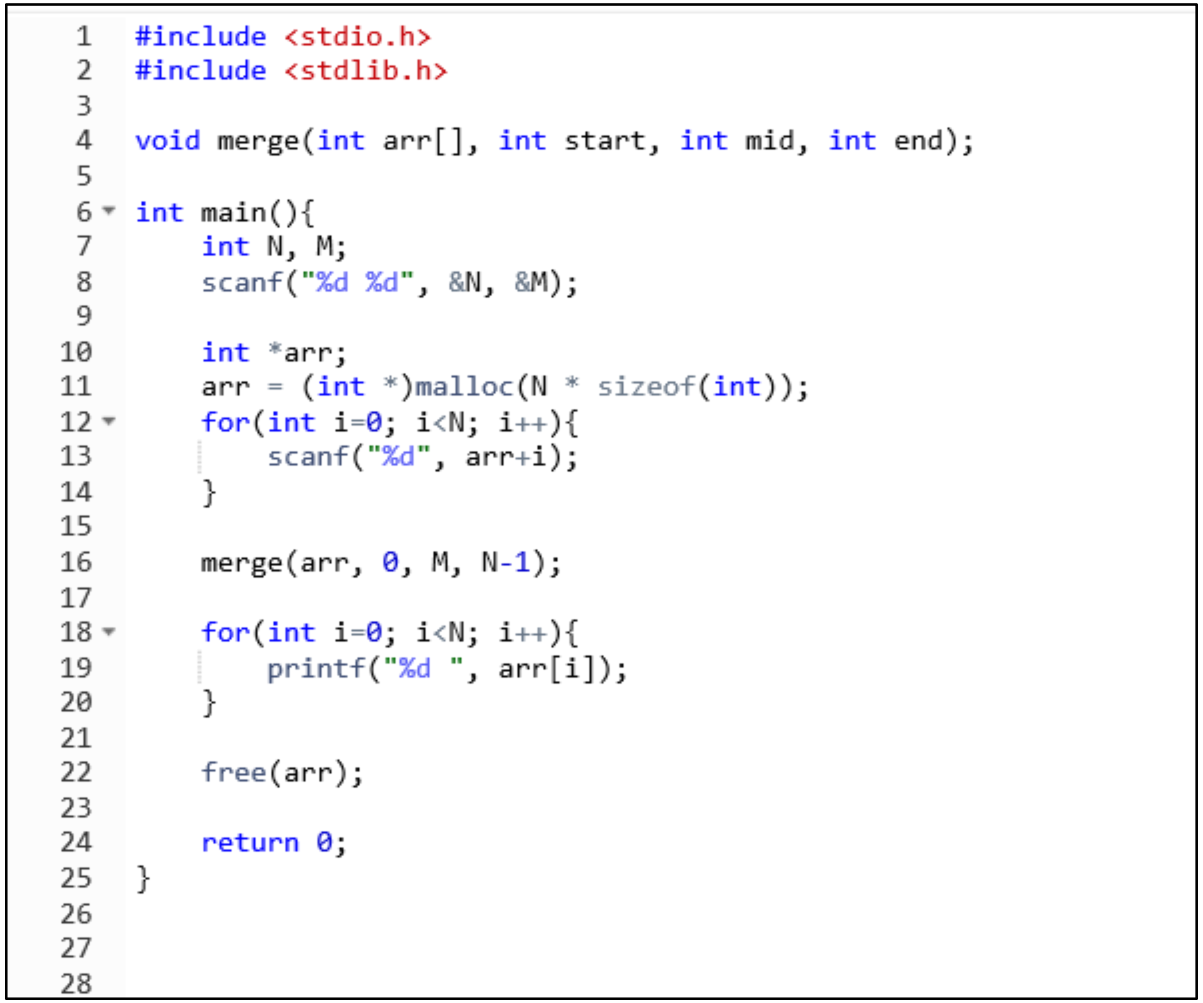
Sample

```
[input]         [input]
6 3             7 2
1 3 5 2 4 6     5 8 1 2 3 4 7
[output]        [output]
1 2 3 4 5 6     1 2 3 4 5 7 8
```

Problem. **3**

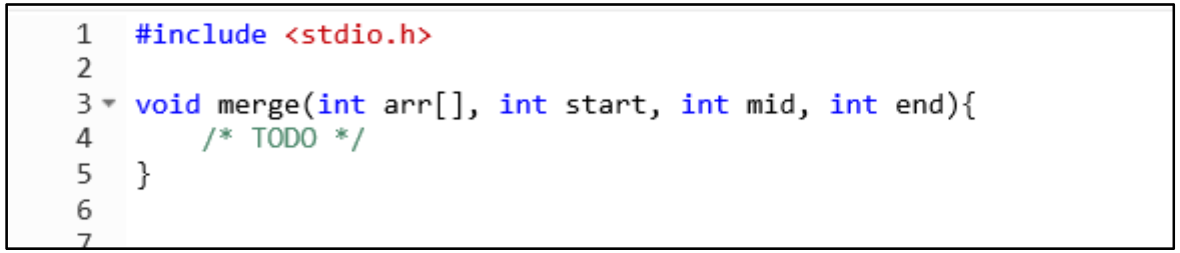
Merging Two Arrays

[main.c]



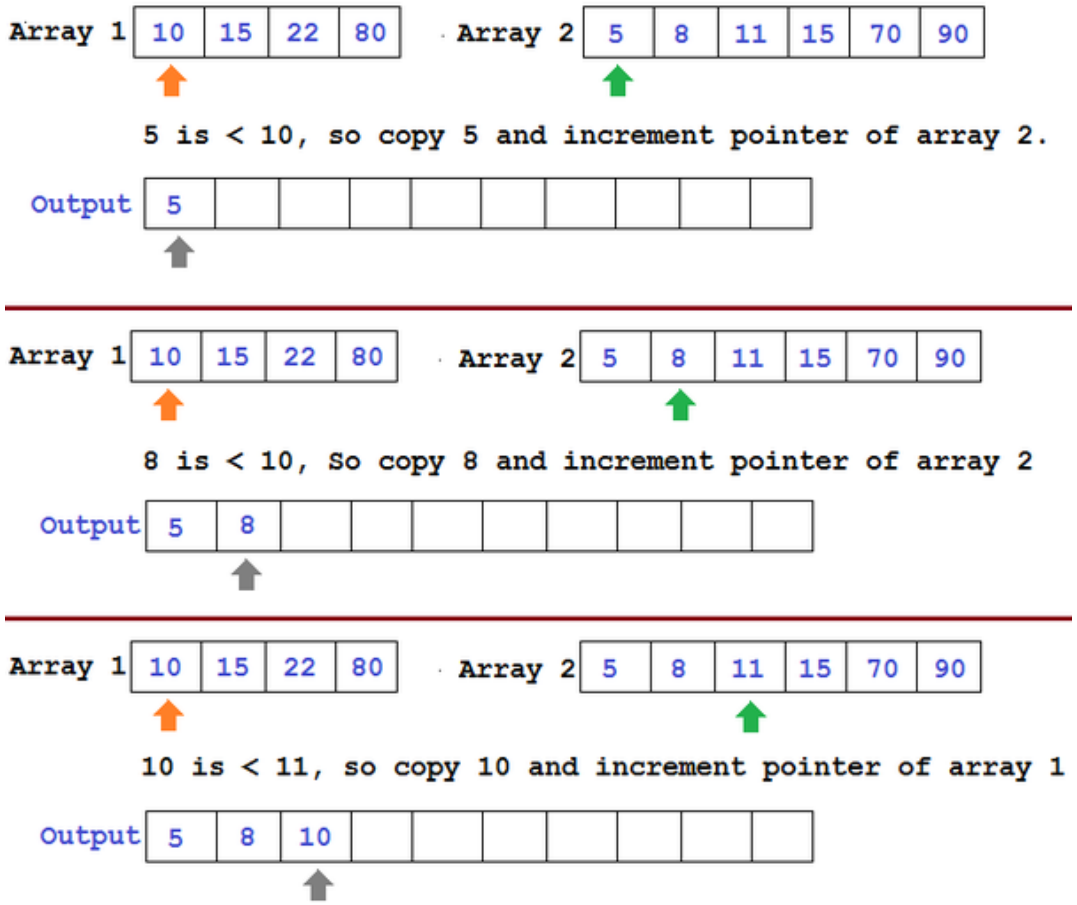
```c
#include <stdio.h>
#include <stdlib.h>

void merge(int arr[], int start, int mid, int end);

int main(){
    int N, M;
    scanf("%d %d", &N, &M);

    int *arr;
    arr = (int *)malloc(N * sizeof(int));
    for(int i=0; i<N; i++){
        scanf("%d", arr+i);
    }

    merge(arr, 0, M, N-1);

    for(int i=0; i<N; i++){
        printf("%d ", arr[i]);
    }

    free(arr);

    return 0;
}
```

[merge.c]



```c
#include <stdio.h>

void merge(int arr[], int start, int mid, int end){
    /* TODO */
}
```

Merging two arrays



Array 1 | 10 | 15 | 22 | 80 | Array 2 | 5 | 8 | 11 | 15 | 70 | 90 |

5 is < 10, so copy 5 and increment pointer of array 2.

Output | 5 | | | | | | | | | |

Array 1 | 10 | 15 | 22 | 80 | Array 2 | 5 | 8 | 11 | 15 | 70 | 90 |

8 is < 10, So copy 8 and increment pointer of array 2

Output | 5 | 8 | | | | | | | | |

Array 1 | 10 | 15 | 22 | 80 | Array 2 | 5 | 8 | 11 | 15 | 70 | 90 |

10 is < 11, so copy 10 and increment pointer of array 1

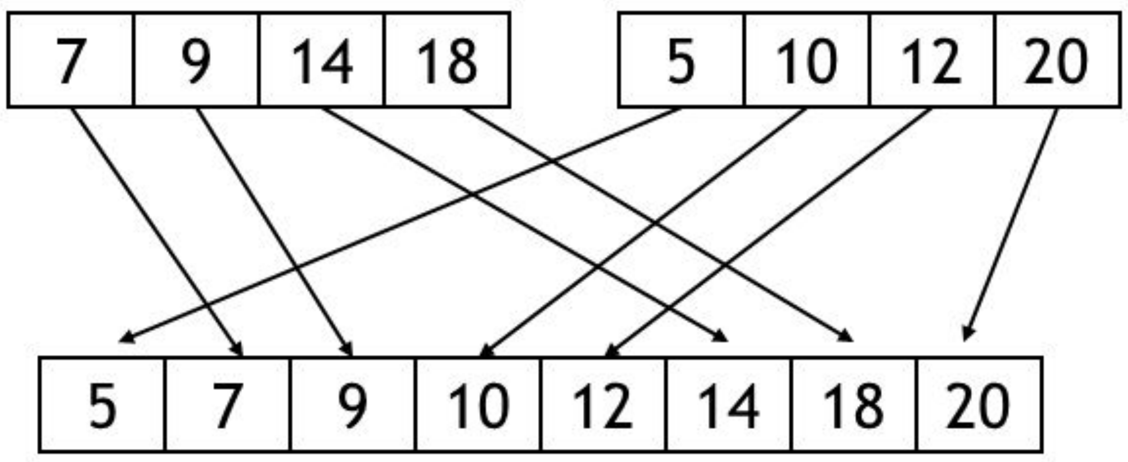
Output | 5 | 8 | 10 | | | | | | | |

Need three index variables:

- Index for Array 1
- Index for Array 2
- Index for (merged) result array

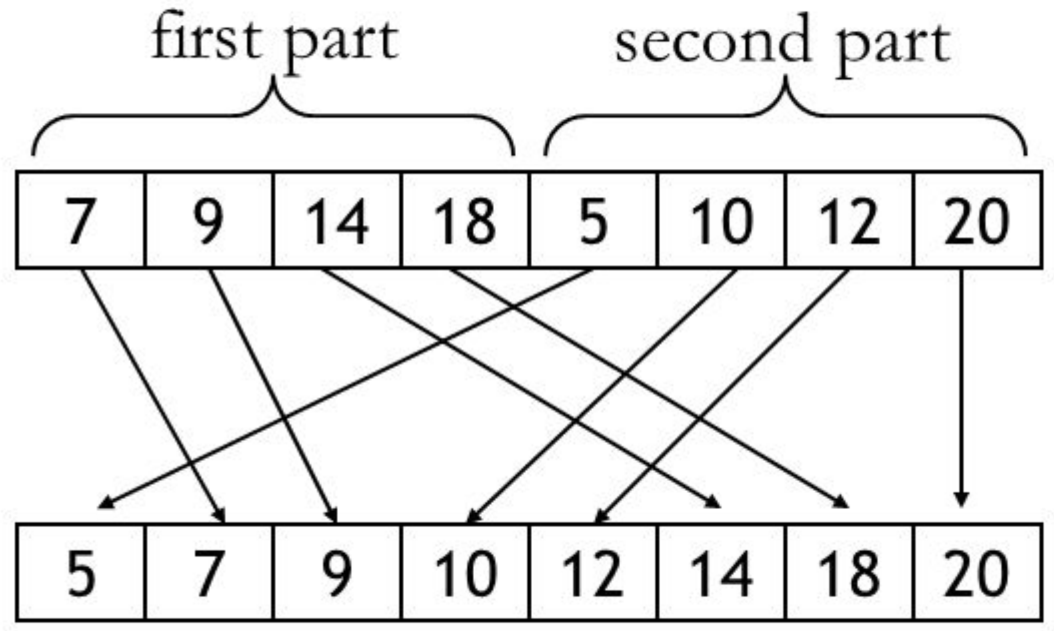
Merging two arrays

- Repeatedly compare the least elements of the two arrays, and copy the smaller one into the third array.

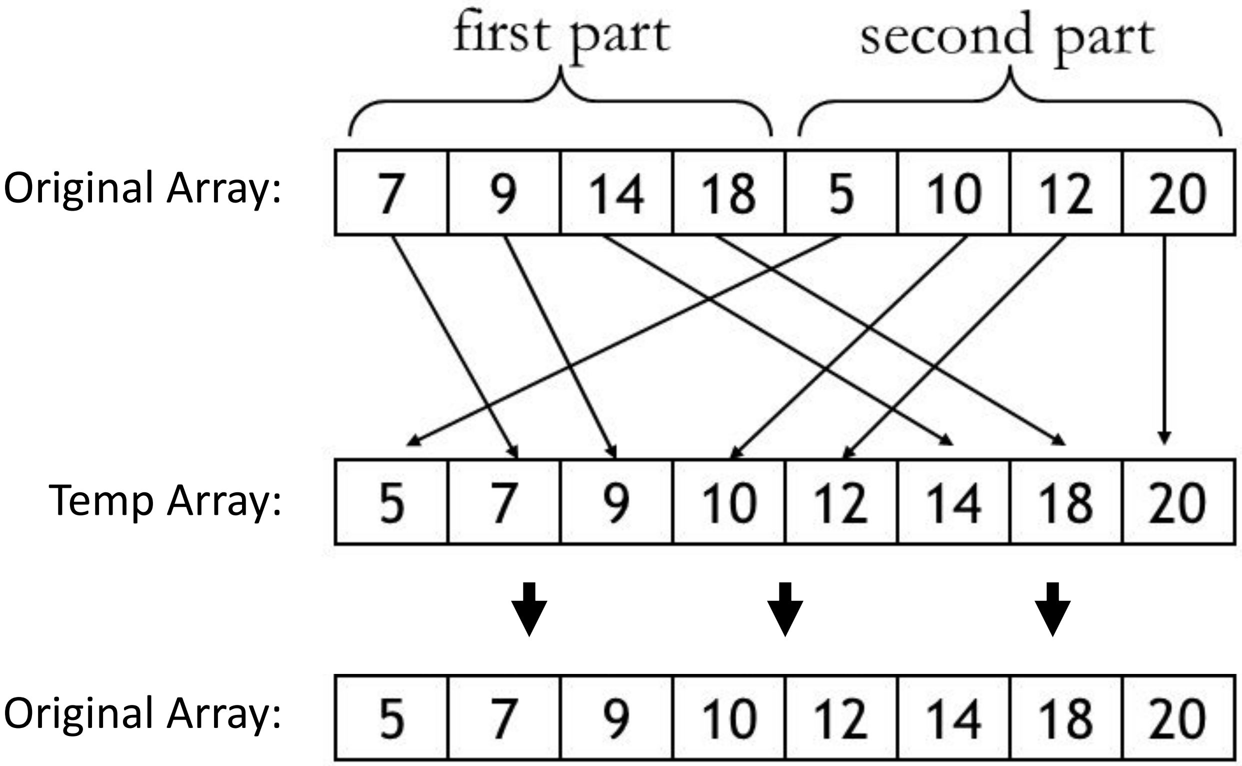


Merging two arrays

- It's no different if the two "arrays" are actually portions of a single array.



Merging two arrays



first part second part

Original Array: 7 9 14 18 5 10 12 20

Temp Array: 5 7 9 10 12 14 18 20

Original Array: 5 7 9 10 12 14 18 20

- Just make sure to first write into a ***temp array***, and then copy back into the ***original array***.

Problem. **4**

Corpus

Description

Several words will be given. Your job is to count how many times each word appears. Then print the word and its count in lexicographical order.

Each word contains at most 50 characters, and the number of given words is at most 500.

Input consists of lowercase alphabets and space only. In other word, every word consists of lowercase alphabets only.

The number of words is not given.

Input

Several words are given in a single line.

Output

Print each word and its count, line by line. (Lexicographical order.)

Sample

```
[input]
i do like you but you do
not like me
[output]
but 1
do 2
i 1
like 2
me 1
not 1
you 2
```