12 Square

12.1 Anonymous Union

As mentioned in the lecture notes, modern C supports anonymous unions. The anonymous union supports the field can be used directly without the union variable. The following code shows an example of an anonymous union:

```c
#include <stdio.h>

typedef struct Price {
    enum {KRW, USD} tag;
    union {
        int    krw;
        double usd;
    };
} Price;

int main() {
    Price price[] = {{KRW, .krw = 3000}, {USD, .usd = 25.99}};
    int size = sizeof price / sizeof *price;

    for (int i = 0; i < size; i++)
        if (price[i].tag == KRW)
            printf("price[%d] = KRW %d\n", i, price[i].krw);
        else
            printf("price[%d] = USD %.2f\n", i, price[i].usd);

    return 0;
}
```

The elements of the array `price` consist of the fields tag and those of unions. To initialize the fields of the anonymous union, we used the dot notation, say `.krw` and `.usd`.

12.2 Programming Lab 12: square.c

Write a program reading n data items and printing the squares of them in reverse. A data item consists of either an integer or a character which is not a numeral nor a sign (+ and −). The square of an integer has the same meaning in mathematics, i.e. the square of 2 is 4 and that of −3 is 9. The square of a character, however, is the repetition of the character in double, i.e. the square of `A` is `AA` and that of `b` is `bb`.

To process the data properly, define and use a tagged structure containing an anonymous union as in the sample code. The type of the structure should be declared as `Data` using `typedef`.

Your program is to read from standard input. Input consists of two lines. The first line of the input contains the number of data items n ($0 < n < 100$). The second line of the input contains n data items separated by space. Your program is to write to standard output. The output consists of a single line containing the squared data items in reverse separated by space.

Additional requirements for bonus points

- Use `switch` statements instead of `if` statements.
- Define and use the function `square(d)` returning the square of the data item `d`. The storage for storing the squared characters should be allocated separately using `malloc`.

Input	**Output**
3 -1 A 5	25 AA 1
4 ^ -4 b +3	9 bb 16 ^^