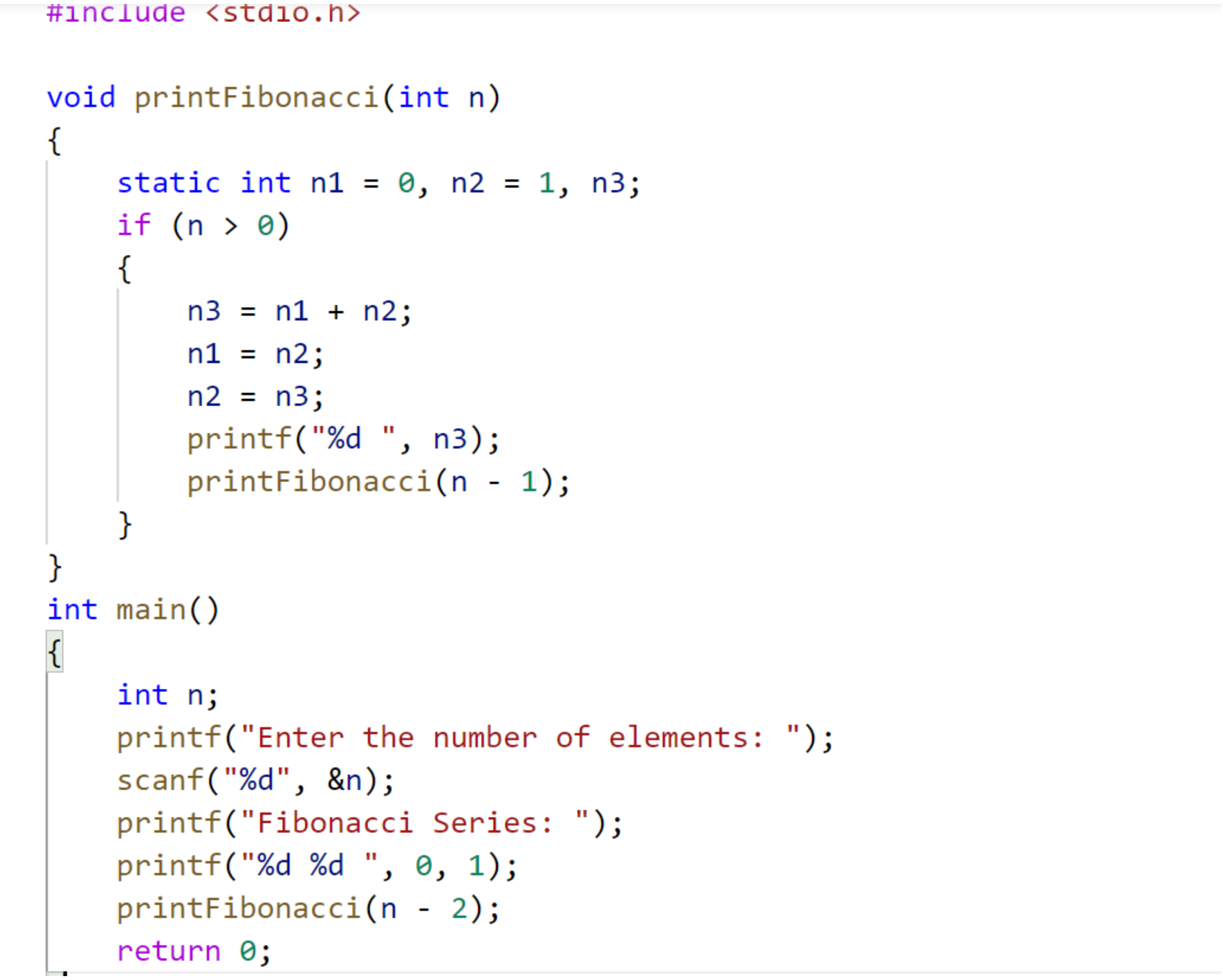
Harsh

201105018

Batch B

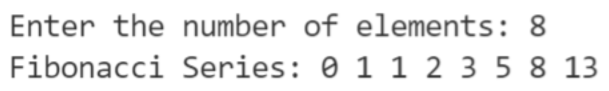
1

Theory: The Fibonacci sequence is a series of numbers where a number is the addition of the last two numbers, starting with 0, and 1. The Fibonacci Sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21



```c
#include <stdio.h>

void printFibonacci(int n)
{
    static int n1 = 0, n2 = 1, n3;
    if (n > 0)
    {
        n3 = n1 + n2;
        n1 = n2;
        n2 = n3;
        printf("%d ", n3);
        printFibonacci(n - 1);
    }
}
int main()
{
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Fibonacci Series: ");
    printf("%d %d ", 0, 1);
    printFibonacci(n - 2);
    return 0;
```

Output:

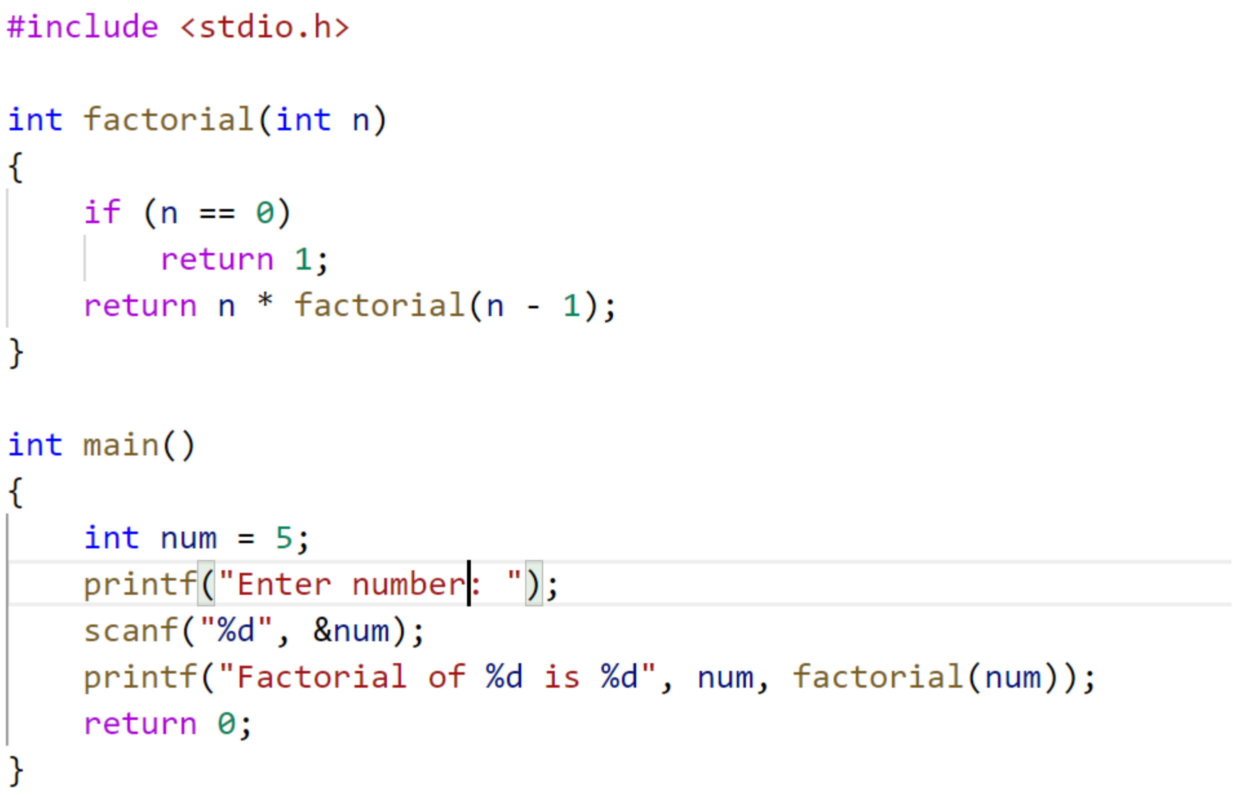


```
Enter the number of elements: 8
Fibonacci Series: 0 1 1 2 3 5 8 13
```

Conclusion: recursion can be easily used to generate Fibonacci series.

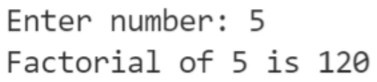
2

Theory: To find the factorial of a number, multiply the number with the factorial value of the previous number.



```c
#include <stdio.h>

int factorial(int n)
{
    if (n == 0)
        return 1;
    return n * factorial(n - 1);
}

int main()
{
    int num = 5;
    printf("Enter number: ");
    scanf("%d", &num);
    printf("Factorial of %d is %d", num, factorial(num));
    return 0;
}
```

Output:

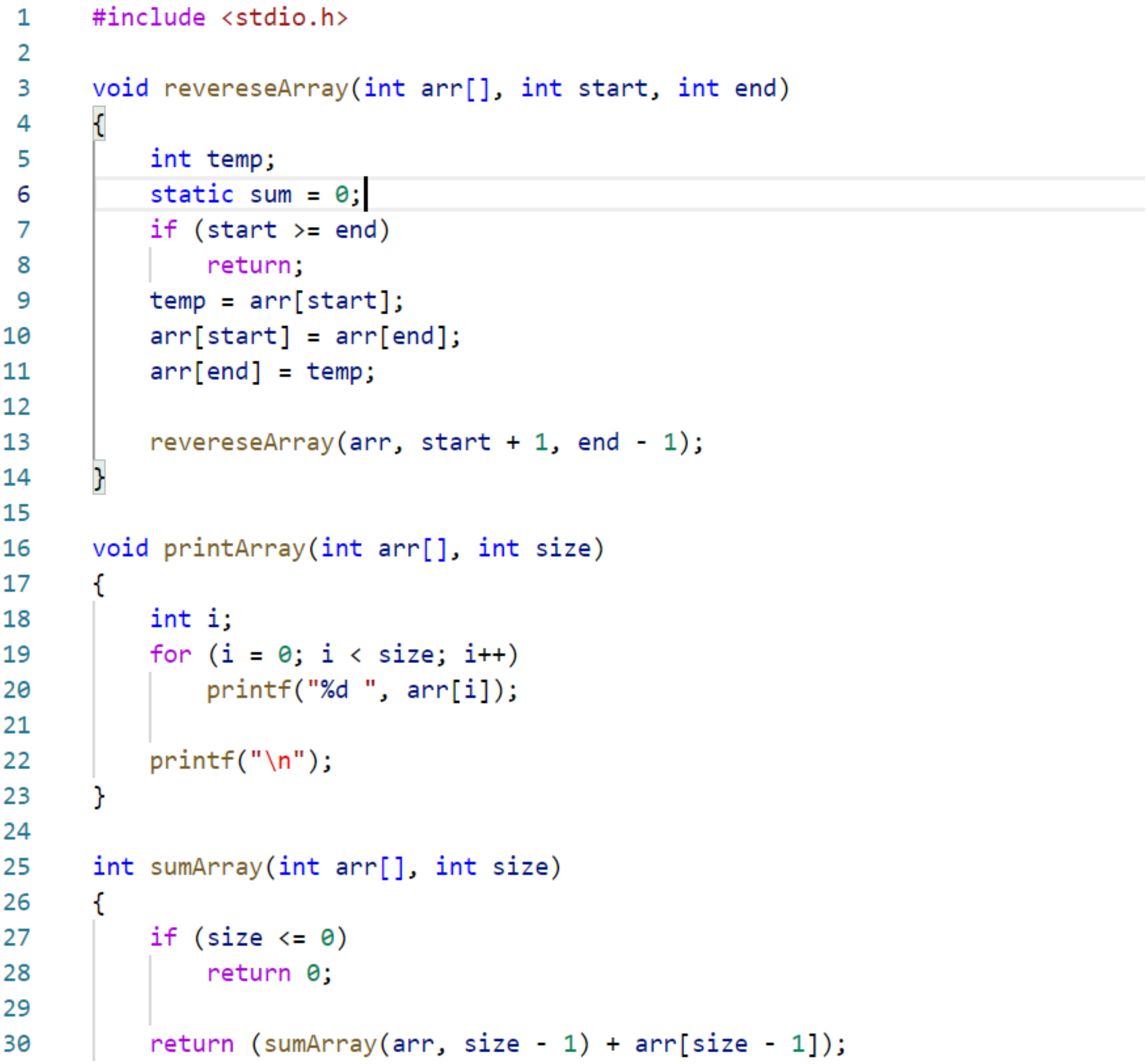


```
Enter number: 5
Factorial of 5 is 120
```

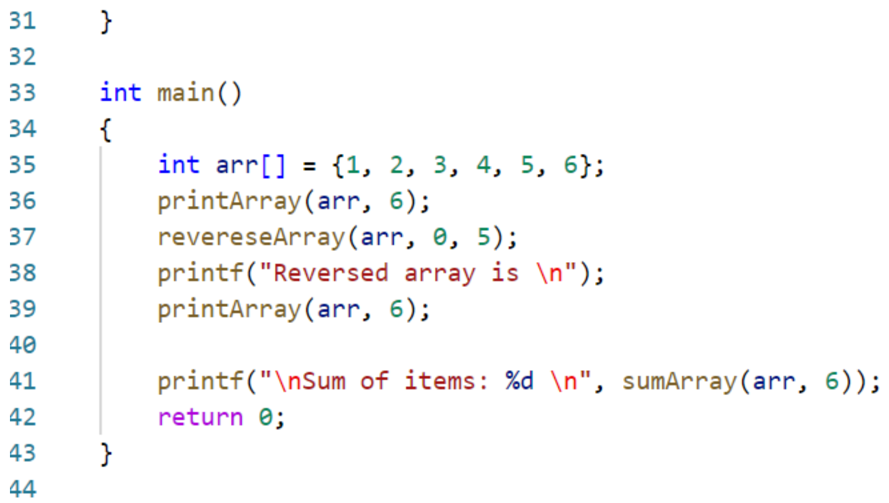
Conclusion: recursion can be easily used to find factorial of a number.

3

Theory: An array is a data structure that contains a group of elements.



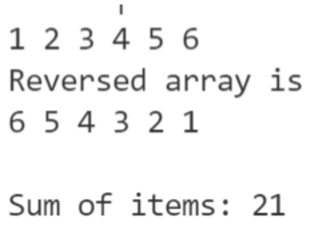
```c
#include <stdio.h>

void revereseArray(int arr[], int start, int end)
{
    int temp;
    static sum = 0;
    if (start >= end)
        return;
    temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;

    revereseArray(arr, start + 1, end - 1);
}

void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);

    printf("\n");
}

int sumArray(int arr[], int size)
{
    if (size <= 0)
        return 0;

    return (sumArray(arr, size - 1) + arr[size - 1]);
```



```c
}

int main()
{
    int arr[] = {1, 2, 3, 4, 5, 6};
    printArray(arr, 6);
    revereseArray(arr, 0, 5);
    printf("Reversed array is \n");
    printArray(arr, 6);

    printf("\nSum of items: %d \n", sumArray(arr, 6));
    return 0;
}
```

Ouput:

Output

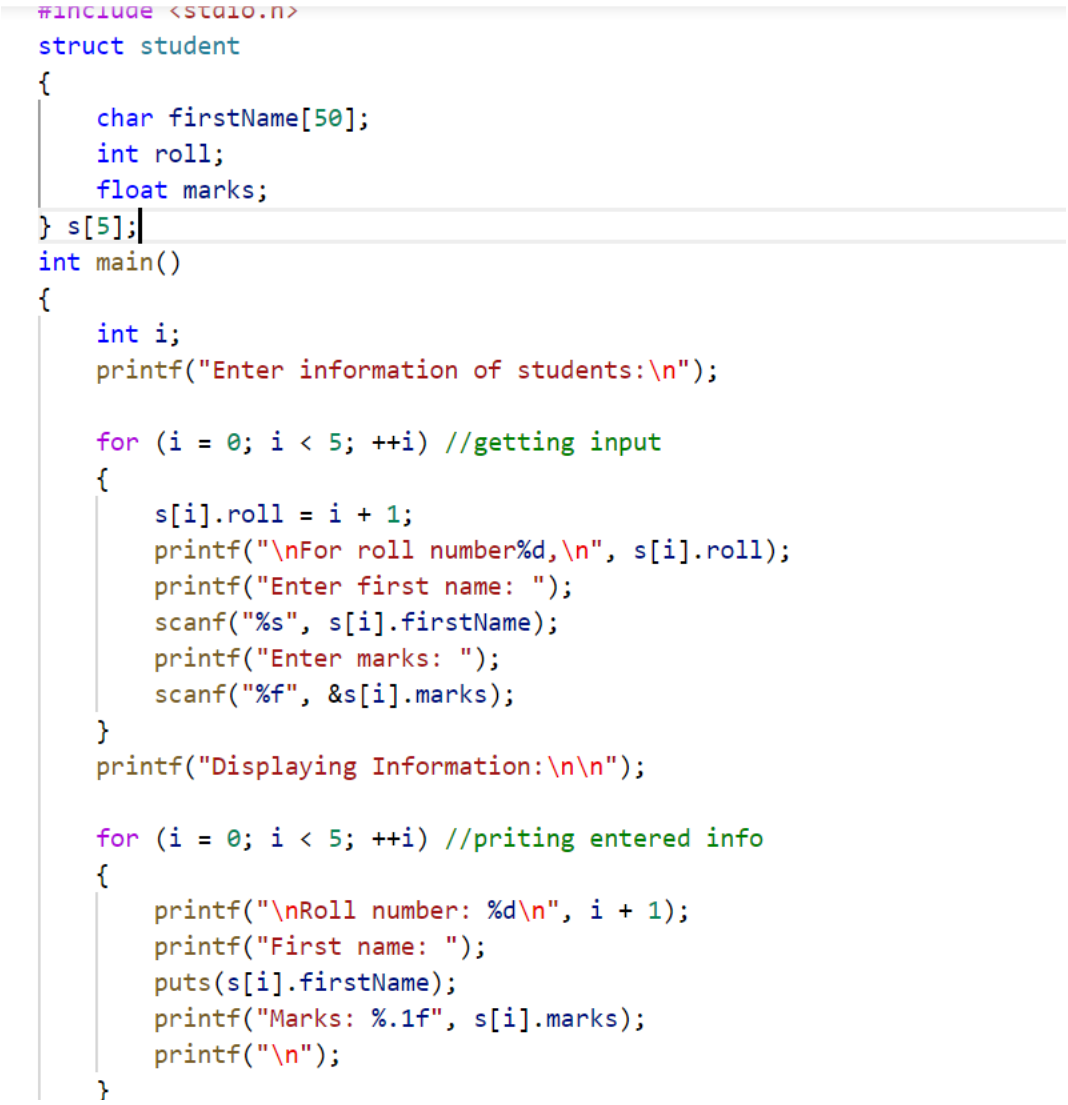


```
1 2 3 4 5 6
Reversed array is
6 5 4 3 2 1

Sum of items: 21
```

Conclusion: recursion can be easily used to reverse an array

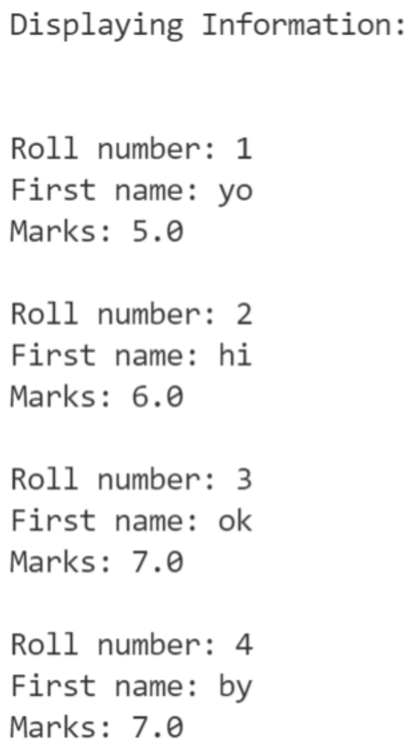
4

Theory: Structures allow us to combine elements of a different data type into a group.



```c
#include <stdio.h>
struct student
{
    char firstName[50];
    int roll;
    float marks;
} s[5];
int main()
{
    int i;
    printf("Enter information of students:\n");

    for (i = 0; i < 5; ++i) //getting input
    {
        s[i].roll = i + 1;
        printf("\nFor roll number%d,\n", s[i].roll);
        printf("Enter first name: ");
        scanf("%s", s[i].firstName);
        printf("Enter marks: ");
        scanf("%f", &s[i].marks);
    }
    printf("Displaying Information:\n\n");

    for (i = 0; i < 5; ++i) //priting entered info
    {
        printf("\nRoll number: %d\n", i + 1);
        printf("First name: ");
        puts(s[i].firstName);
        printf("Marks: %.1f", s[i].marks);
        printf("\n");
    }
```

Output:

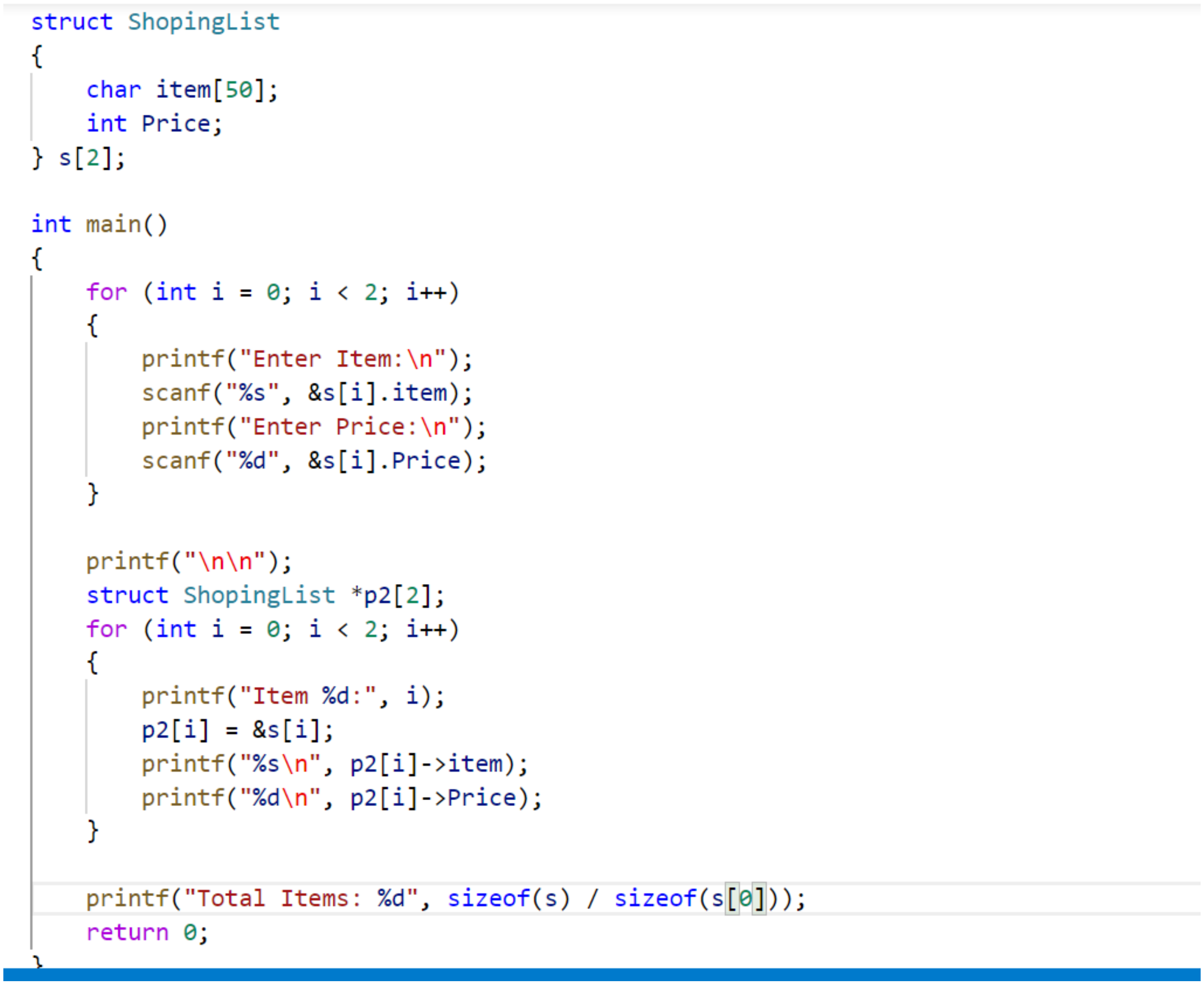


```
Displaying Information:


Roll number: 1
First name: yo
Marks: 5.0

Roll number: 2
First name: hi
Marks: 6.0

Roll number: 3
First name: ok
Marks: 7.0

Roll number: 4
First name: by
Marks: 7.0
```

Conclusion: Structures make code maintainable

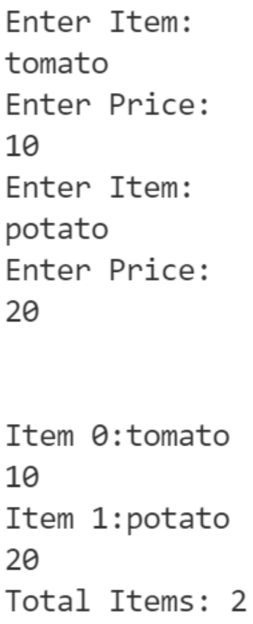
5

Theory: Structures allow us to combine elements of a different data type into a group.



```
struct ShopingList
{
    char item[50];
    int Price;
} s[2];

int main()
{
    for (int i = 0; i < 2; i++)
    {
        printf("Enter Item:\n");
        scanf("%s", &s[i].item);
        printf("Enter Price:\n");
        scanf("%d", &s[i].Price);
    }

    printf("\n\n");
    struct ShopingList *p2[2];
    for (int i = 0; i < 2; i++)
    {
        printf("Item %d:", i);
        p2[i] = &s[i];
        printf("%s\n", p2[i]->item);
        printf("%d\n", p2[i]->Price);
    }

    printf("Total Items: %d", sizeof(s) / sizeof(s[0]));
    return 0;
}
```

Output:



```
Enter Item:
tomato
Enter Price:
10
Enter Item:
potato
Enter Price:
20


Item 0:tomato
10
Item 1:potato
20
Total Items: 2
```

Conclusion: Points can be easily used for structures like any other data type.