GE23131-Programming Using C-2024-2025

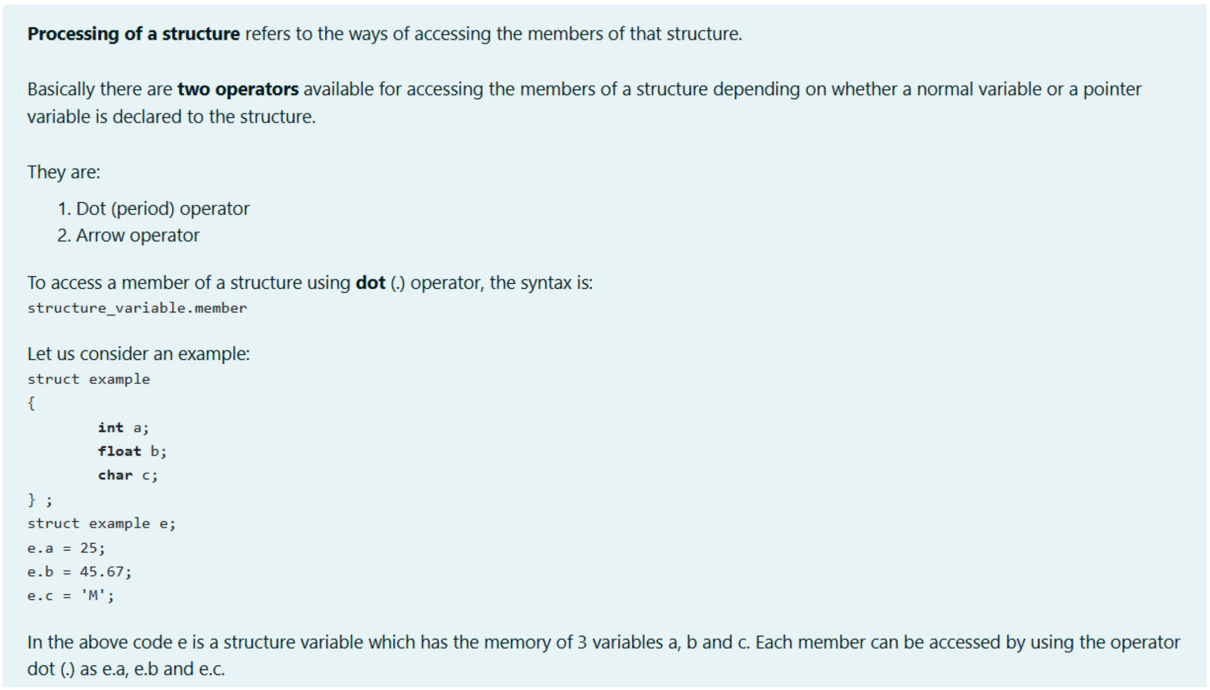
Week 4 Assessment

Abenanthan P

240701005

Decision Making and Branching- if , if…else , if..else if

Initialization of Structures and Accessing Structures



Processing of a structure refers to the ways of accessing the members of that structure.

Basically there are **two operators** available for accessing the members of a structure depending on whether a normal variable or a pointer variable is declared to the structure.

They are:

1. Dot (period) operator
2. Arrow operator

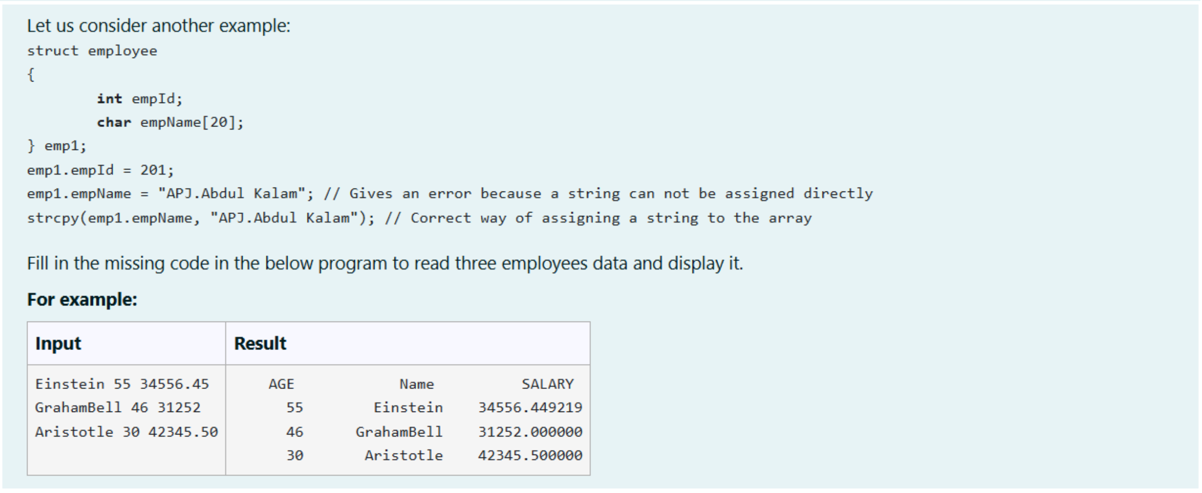
To access a member of a structure using **dot** (.) operator, the syntax is:

```
structure_variable.member
```

Let us consider an example:

```
struct example
{
        int a;
        float b;
        char c;
} ;
struct example e;
e.a = 25;
e.b = 45.67;
e.c = 'M';
```

In the above code e is a structure variable which has the memory of 3 variables a, b and c. Each member can be accessed by using the operator dot (.) as e.a, e.b and e.c.



Let us consider another example:

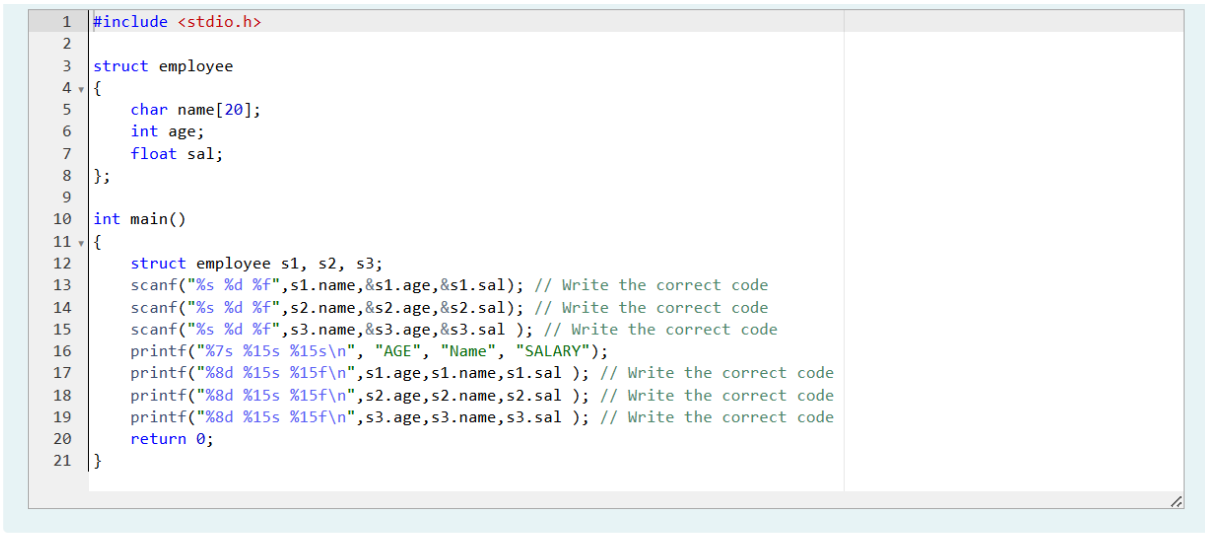
```
struct employee
{
        int empId;
        char empName[20];
} emp1;
emp1.empId = 201;
emp1.empName = "APJ.Abdul Kalam"; // Gives an error because a string can not be assigned directly
strcpy(emp1.empName, "APJ.Abdul Kalam"); // Correct way of assigning a string to the array
```

Fill in the missing code in the below program to read three employees data and display it.

For example:

Input	Result

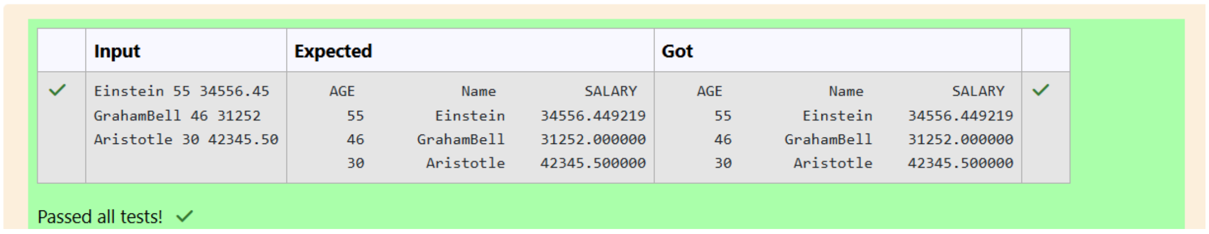
Einstein 55 34556.45
GrahamBell 46 31252
Aristotle 30 42345.50
	`   AGE          Name        SALARY`
	`    55      Einstein  34556.449219`
	`    46    GrahamBell  31252.000000`
	`    30     Aristotle  42345.500000`



```c
#include <stdio.h>

struct employee
{
    char name[20];
    int age;
    float sal;
};

int main()
{
    struct employee s1, s2, s3;
    scanf("%s %d %f",s1.name,&s1.age,&s1.sal); // Write the correct code
    scanf("%s %d %f",s2.name,&s2.age,&s2.sal); // Write the correct code
    scanf("%s %d %f",s3.name,&s3.age,&s3.sal ); // Write the correct code
    printf("%7s %15s %15s\n", "AGE", "Name", "SALARY");
    printf("%8d %15s %15f\n",s1.age,s1.name,s1.sal ); // Write the correct code
    printf("%8d %15s %15f\n",s2.age,s2.name,s2.sal ); // Write the correct code
    printf("%8d %15s %15f\n",s3.age,s3.name,s3.sal ); // Write the correct code
    return 0;
}
```



	Input	Expected	Got	

✓	`Einstein 55 34556.45`
	`GrahamBell 46 31252`
	`Aristotle 30 42345.50`
		`    AGE            Name          SALARY`
		`      55        Einstein    34556.449219`
		`      46      GrahamBell    31252.000000`
		`      30       Aristotle    42345.500000`
			`    AGE            Name          SALARY`
			`      55        Einstein    34556.449219`
			`      46      GrahamBell    31252.000000`
			`      30       Aristotle    42345.500000`
				✓

Passed all tests! ✓

Initialization of Structures and Accessing Structures

A **structure variable** can be initialized in its definition itself with a list of initializers enclosed within the braces. The initialization of values will be taken in an order.

Let us consider an example:

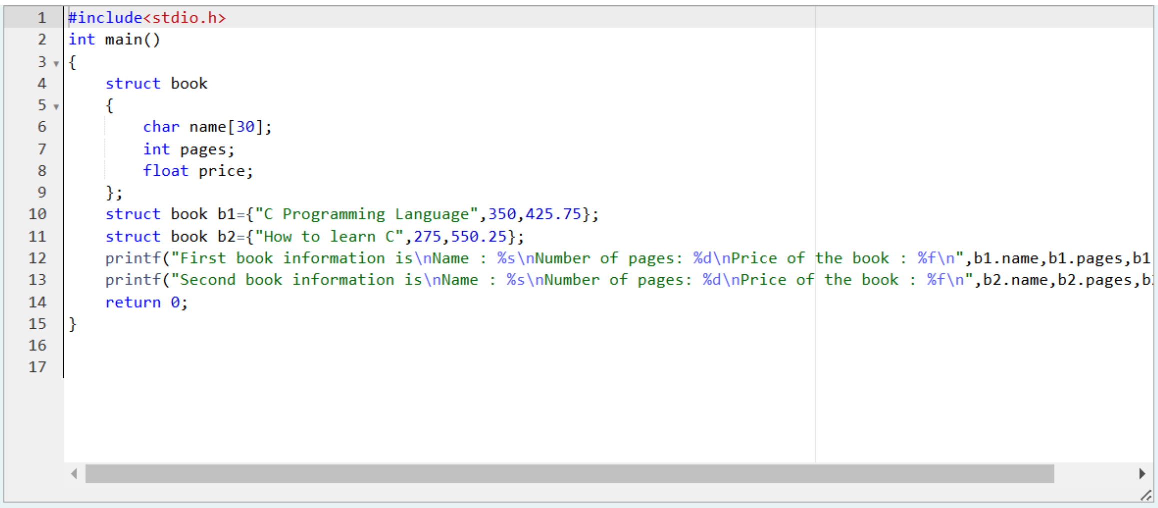
```c
struct student
{
        int roll_number;
        int total_marks;
        float attendance_percentage;
} ;
struct student s = {101, 450, 72.45};
```

In the above example 101 is assigned to **s.roll_number**, 450 is assigned to **s.total_marks** and 72.45 is assigned to **s.attendance_percentage**.

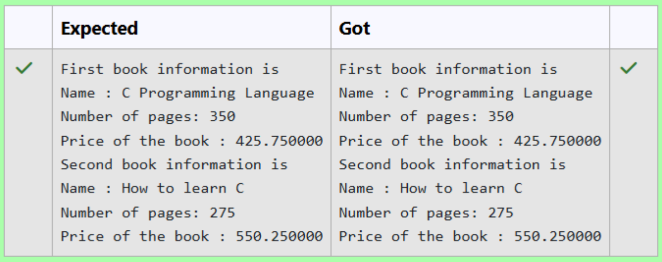
See and retype the below code.

```c
#include <stdio.h>

int main()
{
   struct book
   {
   char name[30];
   int pages;
   float price;
   };
   struct book b1 = {"C Programming Language", 350, 425.75};
   struct book b2 = {"How to learn C", 275, 550.25};
   printf("First book information is\nName : %s\nNumber of pages: %d\nPrice of the book : %f\n", b1.name, b1.pages, b1.price);
   printf("Second book information is\nName : %s\nNumber of pages: %d\nPrice of the book : %f\n", b2.name, b2.pages, b2.price);
   return 0;
}
```



```c
#include<stdio.h>
int main()
{
    struct book
    {
        char name[30];
        int pages;
        float price;
    };
    struct book b1={"C Programming Language",350,425.75};
    struct book b2={"How to learn C",275,550.25};
    printf("First book information is\nName : %s\nNumber of pages: %d\nPrice of the book : %f\n",b1.name,b1.pages,b1
    printf("Second book information is\nName : %s\nNumber of pages: %d\nPrice of the book : %f\n",b2.name,b2.pages,b
    return 0;
}
```



	Expected	Got	
✓	First book information is Name : C Programming Language Number of pages: 350 Price of the book : 425.750000 Second book information is Name : How to learn C Number of pages: 275 Price of the book : 550.250000	First book information is Name : C Programming Language Number of pages: 350 Price of the book : 425.750000 Second book information is Name : How to learn C Number of pages: 275 Price of the book : 550.250000	✓

Nested Structures and Array of Structures

When the programmer wants to define more number of structure variables, it is better practice to define **array of structures** rather than individual structure variables.

Creation of **array of structures** is same as creation of **integer arrays**, **float arrays** and so on.

The syntax of defining an array of structure is: Below is an example of array of structure:

```c
struct tag_name structure_variable[size]; // The size specifies the number of structure variables to be created
```

Below is an example of array of structure:

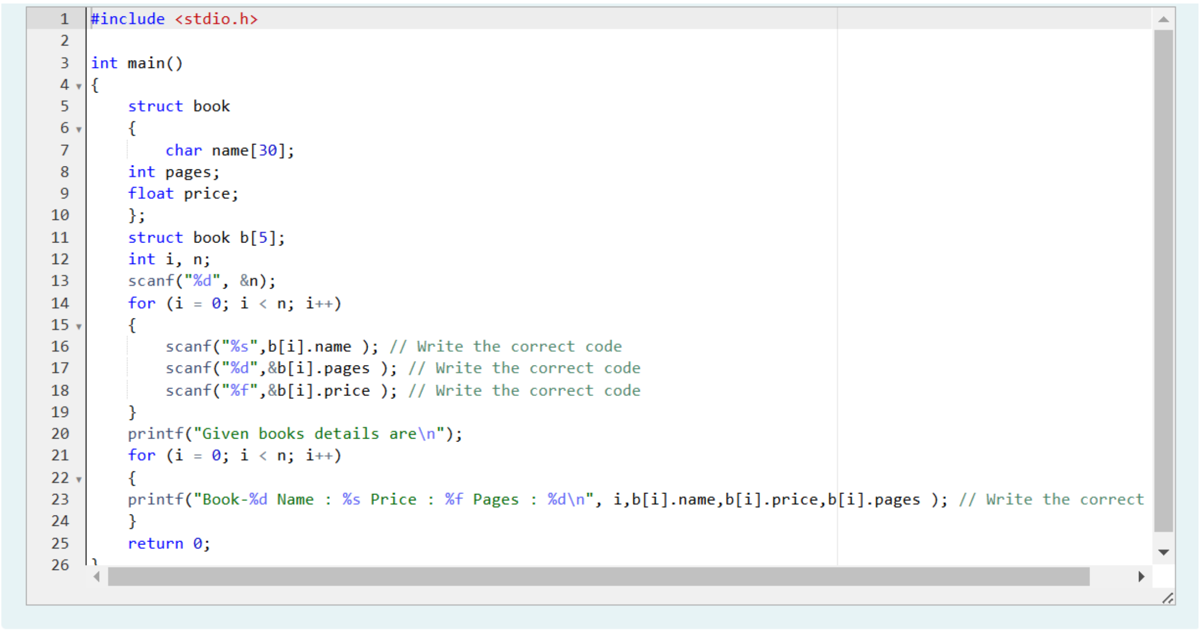
```c
struct example
{
        int a;
        float b;
};
struct example e[3];
```

In the above example the definition of a structure will create an array of structures with **3** structure variables of type **structure example**. An array of structure variables should get a memory of continuous blocks and they can be accessed sequentially.

e[**0**].a e[**0**].b e[**1**].a e[**1**].b e[**2**].a e[**2**].b

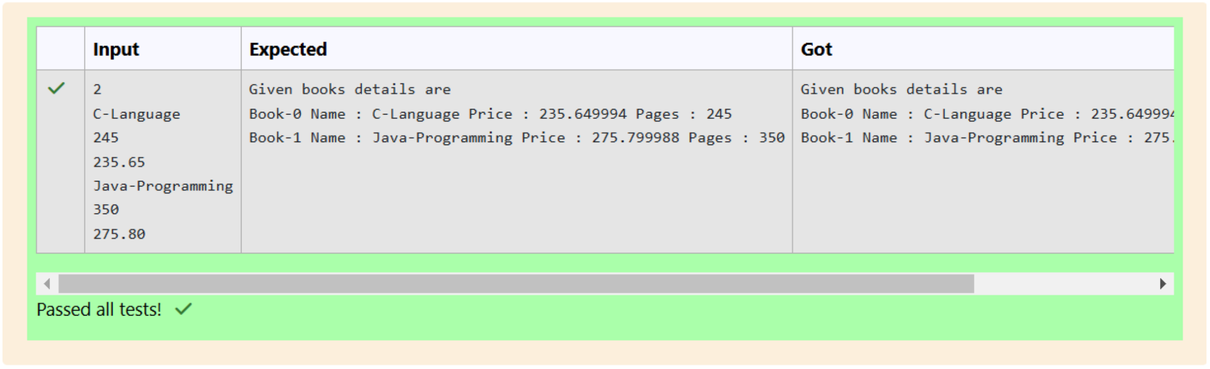
4724 4726 4730 4732 4736 4738

Understand the below program and then fill in the missing code to read **n** books data and display it.



```c
#include <stdio.h>

int main()
{
    struct book
    {
        char name[30];
    int pages;
    float price;
    };
    struct book b[5];
    int i, n;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%s",b[i].name ); // Write the correct code
        scanf("%d",&b[i].pages ); // Write the correct code
        scanf("%f",&b[i].price ); // Write the correct code
    }
    printf("Given books details are\n");
    for (i = 0; i < n; i++)
    {
    printf("Book-%d Name : %s Price : %f Pages : %d\n", i,b[i].name,b[i].price,b[i].pages ); // Write the correct
    }
    return 0;
}
```



	Input	Expected	Got
✓	2 C-Language 245 235.65 Java-Programming 350 275.80	Given books details are Book-0 Name : C-Language Price : 235.649994 Pages : 245 Book-1 Name : Java-Programming Price : 275.799988 Pages : 350	Given books details are Book-0 Name : C-Language Price : 235.64999 Book-1 Name : Java-Programming Price : 275.

Passed all tests! ✓

Nested Structures and Array of Structures

Write a **C** program to find out the total and average marks gained by the students in a section using **array of structures**.

Note: Consider that regdno, marks of 3 subjects, total and average are the members of a structure and make sure to provide the int value for **number of students** which are less than 60

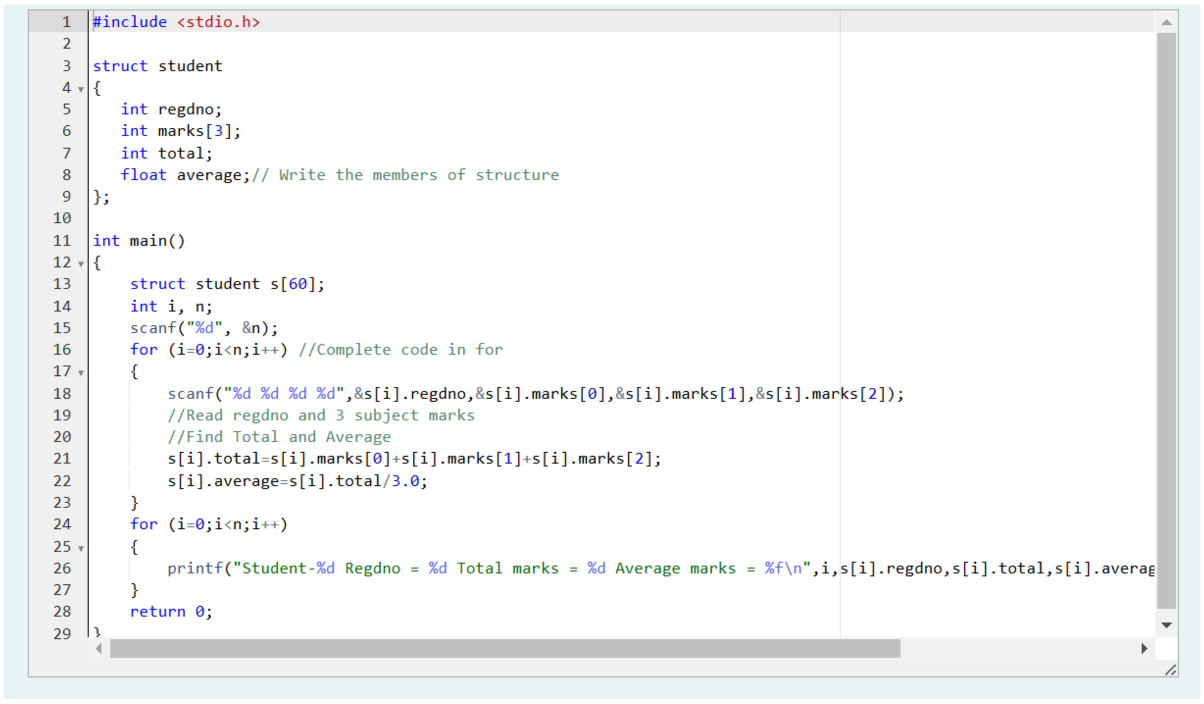
Sample Input and Output:

```
3
101 56 78 76
201 76 89 91
301 46 57 61
Student-0 Regdno = 101 Total marks = 210 Average marks = 70.000000
Student-1 Regdno = 201 Total marks = 256 Average marks = 85.333336
Student-2 Regdno = 301 Total marks = 164 Average marks = 54.666668
```

For example:

Input	Result

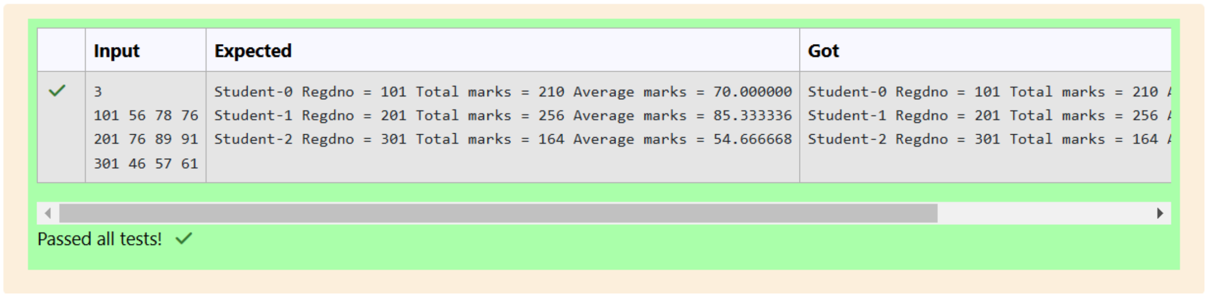
3
101 56 78 76
201 76 89 91
301 46 57 61
	Student-0 Regdno = 101 Total marks = 210 Average marks = 70.000000
	Student-1 Regdno = 201 Total marks = 256 Average marks = 85.333336
	Student-2 Regdno = 301 Total marks = 164 Average marks = 54.666668



```c
#include <stdio.h>

struct student
{
    int regdno;
    int marks[3];
    int total;
    float average;// Write the members of structure
};

int main()
{
    struct student s[60];
    int i, n;
    scanf("%d", &n);
    for (i=0;i<n;i++) //Complete code in for
    {
        scanf("%d %d %d %d",&s[i].regdno,&s[i].marks[0],&s[i].marks[1],&s[i].marks[2]);
        //Read regdno and 3 subject marks
        //Find Total and Average
        s[i].total=s[i].marks[0]+s[i].marks[1]+s[i].marks[2];
        s[i].average=s[i].total/3.0;
    }
    for (i=0;i<n;i++)
    {
        printf("Student-%d Regdno = %d Total marks = %d Average marks = %f\n",i,s[i].regdno,s[i].total,s[i].averag
    }
    return 0;
}
```



	Input	Expected	Got
✓	3 101 56 78 76 201 76 89 91 301 46 57 61	Student-0 Regdno = 101 Total marks = 210 Average marks = 70.000000 Student-1 Regdno = 201 Total marks = 256 Average marks = 85.333336 Student-2 Regdno = 301 Total marks = 164 Average marks = 54.666668	Student-0 Regdno = 101 Total marks = 210 Student-1 Regdno = 201 Total marks = 256 Student-2 Regdno = 301 Total marks = 164

Passed all tests! ✓

Calculate Grade

Write a program that accepts the marks in 3 subjects of a student, calculates the average mark of the student and prints the student's grade. If the average mark is greater than or equal to 90, then the grade is 'A'. If the average mark is 80 and between 80 and 90, then the grade is 'B'. If the average mark is 70 and between 70 and 80, then the grade is 'C'. If the average mark is 60 and between 60 and 70, then the grade is 'D'. If the average mark is 50 and between 50 and 60, then the grade is 'E'. If the average mark is less than 50, then the grade is 'F'.

Input Format:

Input consists of 3 lines. Each line consists of an integer.

Output Format:

Output consists of a single line. Refer sample output for the format.

Sample Input 1 :

```
45
45
45
```

Sample Output 1 :

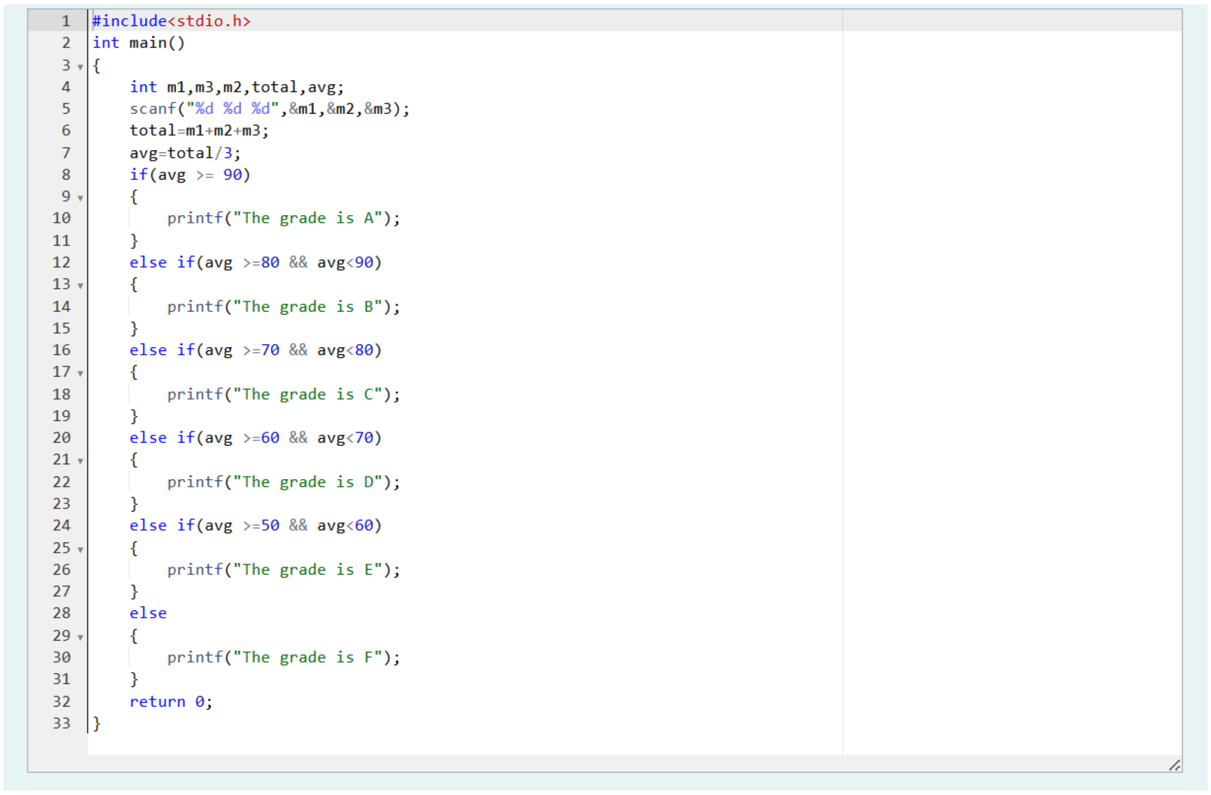
```
The grade is F
```

Sample Input 2:

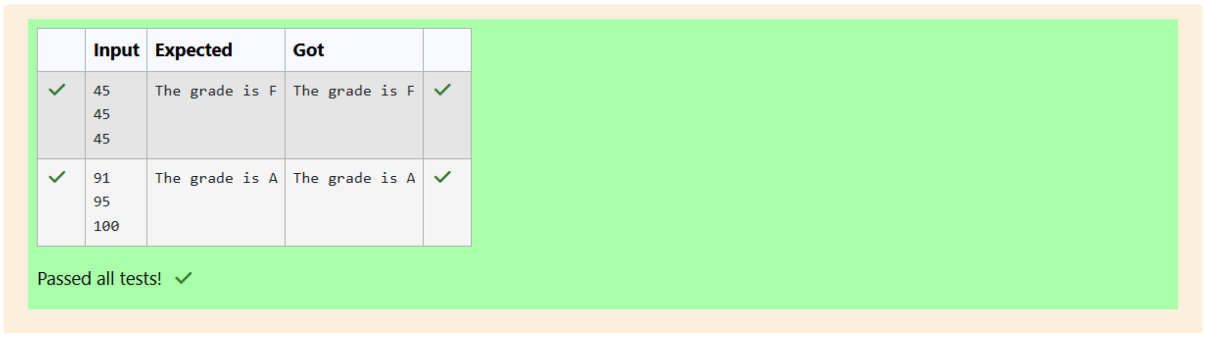
```
91
95
100
```

Sample Output 2:

```
The grade is A
```



```c
#include<stdio.h>
int main()
{
    int m1,m3,m2,total,avg;
    scanf("%d %d %d",&m1,&m2,&m3);
    total=m1+m2+m3;
    avg=total/3;
    if(avg >= 90)
    {
        printf("The grade is A");
    }
    else if(avg >=80 && avg<90)
    {
        printf("The grade is B");
    }
    else if(avg >=70 && avg<80)
    {
        printf("The grade is C");
    }
    else if(avg >=60 && avg<70)
    {
        printf("The grade is D");
    }
    else if(avg >=50 && avg<60)
    {
        printf("The grade is E");
    }
    else
    {
        printf("The grade is F");
    }
    return 0;
}
```



	Input	Expected	Got	
✓	45 45 45	The grade is F	The grade is F	✓
✓	91 95 100	The grade is A	The grade is A	✓

Passed all tests! ✓

Railway Seating Arrangement

Write a program to determine the type of berth when the seat / berth number in the train is given.

Input Format:

Input consists of a single integer. Assume that the range of input is between 1 and 72.

Output Format:

Output consists of a single string. [Upper or Middle or Lower or Side Lower or Side Upper]

Sample Input 1:

9

Sample Output 1:

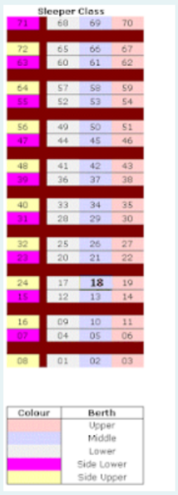
Lower

Sample Input 2:

72

Sample Output 2:

Side Upper



For example:

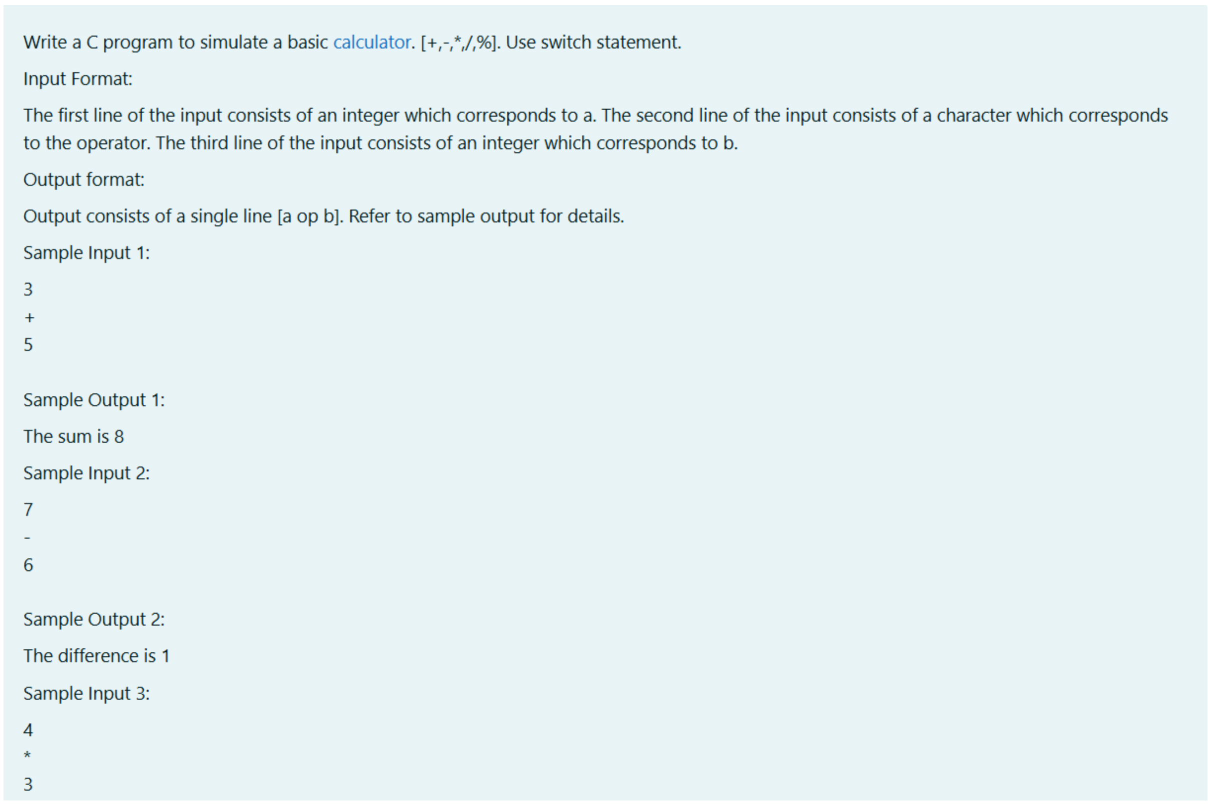
Input	Result
9	Lower
72	Side Upper

```
#include<stdio.h>
int main()
{
    int pos;
    scanf("%d",&pos);
    if (pos==1||pos==4||pos==9||pos==12||pos==17||pos==20||pos==25||pos==28||pos==33||pos==36||pos==41||pos==44||p
    {
        printf("Lower\n");
    }
    else if(pos==2||pos==5||pos==10||pos==13||pos==18||pos==21||pos==26||pos==29||pos==34||pos==37||pos==42||pos==
    {
        printf("Middle\n");
    }
    else if(pos==3||pos==6||pos==11||pos==14||pos==19||pos==22||pos==27||pos==30||pos==35||pos==38||pos==43||pos==
    {
        printf("Upper\n");
    }
    else if(pos==7||pos==15||pos==23||pos==31||pos==39||pos==47||pos==55||pos==63||pos==71)
    {
        printf("Side Lower\n");
    }
    else
    {
        printf("Side Upper\n");
    }
}
```

	Input	Expected	Got	
✓	9	Lower	Lower	✓
✓	72	Side Upper	Side Upper	✓

Passed all tests! ✓

Basic Calculator



Write a C program to simulate a basic calculator. [+,-,*,/,%]. Use switch statement.

Input Format:

The first line of the input consists of an integer which corresponds to a. The second line of the input consists of a character which corresponds to the operator. The third line of the input consists of an integer which corresponds to b.

Output format:

Output consists of a single line [a op b]. Refer to sample output for details.

Sample Input 1:

```
3
+
5
```

Sample Output 1:

```
The sum is 8
```

Sample Input 2:

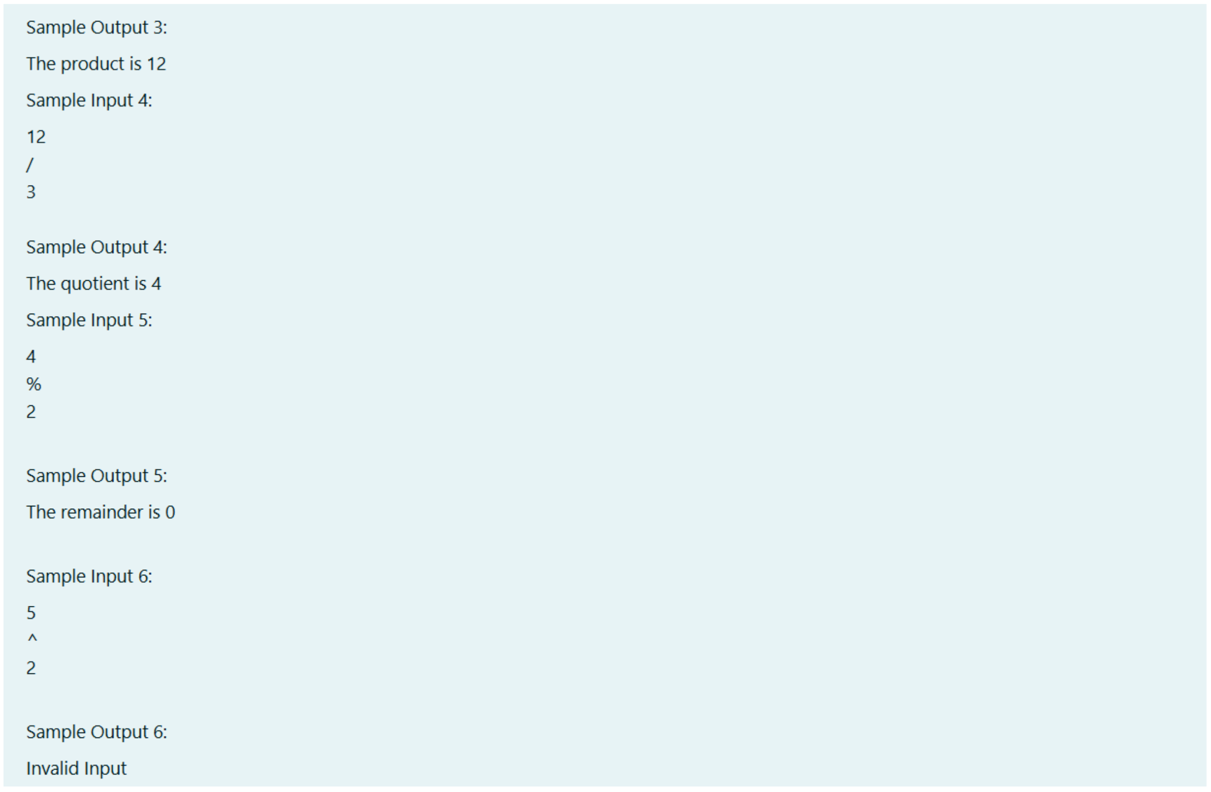
```
7
-
6
```

Sample Output 2:

```
The difference is 1
```

Sample Input 3:

```
4
*
3
```



Sample Output 3:

```
The product is 12
```

Sample Input 4:

```
12
/
3
```

Sample Output 4:

```
The quotient is 4
```

Sample Input 5:

```
4
%
2
```

Sample Output 5:

```
The remainder is 0
```

Sample Input 6:

```
5
^
2
```

Sample Output 6:

```
Invalid Input
```

```c
#include<stdio.h>
int main()
{
    int a,b;
    char op;
    scanf("%d",&a);
    getchar();
    scanf("%c",&op);
    scanf("%d",&b);
    switch(op)
    {
        case'+':
        printf("The sum is %d",a+b);
        break;
        case'-':
        printf("The difference is %d",a-b);
        break;
        case'*':
        printf("The product is %d",a*b);
        break;
        case'/':
        printf("The quotient is %d",a/b);
        break;
        case'%':
        printf("The remainder is %d",a%b);
        break;
        default:
        printf("Invalid Input");
        break;
    }
    return 0;
}
```

	Input	Expected	Got	
✓	3 + 5	The sum is 8	The sum is 8	✓
✓	7 - 6	The difference is 1	The difference is 1	✓
✓	4 * 3	The product is 12	The product is 12	✓
✓	12 / 3	The quotient is 4	The quotient is 4	✓
✓	4 % 2	The remainder is 0	The remainder is 0	✓
✓	5 ^ 2	Invalid Input	Invalid Input	✓

Passed all tests! ✓

Doll Show

In london, every year during dasara there will be a very grand doll show. People try to invent new new dolls of different varieties. The best sold doll's creator will be awarded with cash prize. So people broke their head to create dolls innovatively. Knowing this competition, Mr.Lokpaul tried to create a doll which sings only when a even number is pressed and the number should be not be zero and greater than 100.

So write a program to help Mr.Lokpaul to win.

Input Format:

Input Consists of Single Integer which Corresponds to Number pressed by the user to the doll.

Output Format:

Display whether the doll will Sing or not. Output consists of the string "Doll will sing" or "Invalid number".

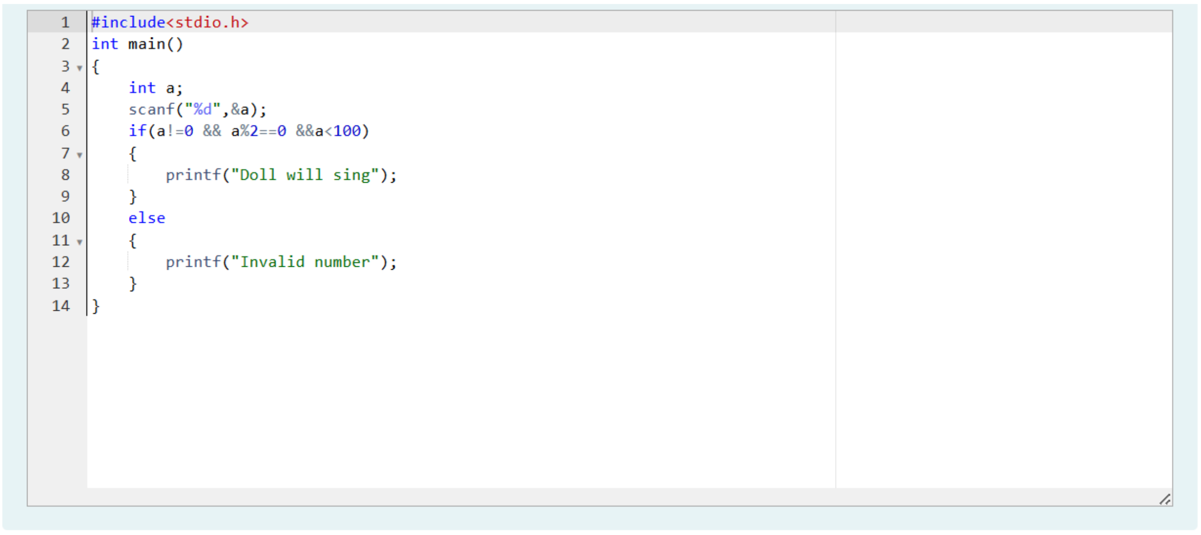
Sample Input and Output:

Input

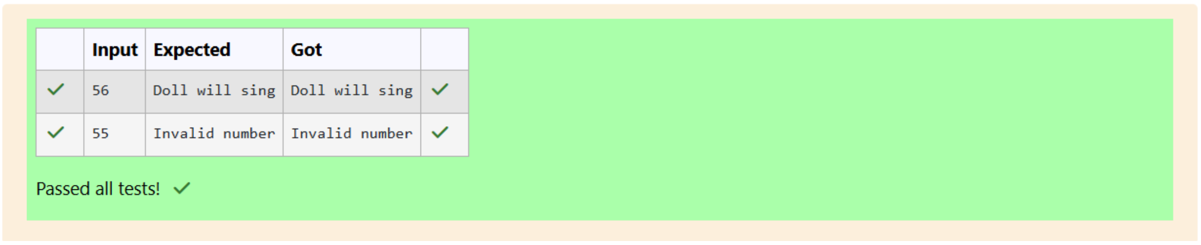
Press a number : 56

Output

Doll will sing



```c
#include<stdio.h>
int main()
{
    int a;
    scanf("%d",&a);
    if(a!=0 && a%2==0 &&a<100)
    {
        printf("Doll will sing");
    }
    else
    {
        printf("Invalid number");
    }
}
```



	Input	Expected	Got	
✓	56	Doll will sing	Doll will sing	✓
✓	55	Invalid number	Invalid number	✓

Passed all tests! ✓