Each Sunday, a newspaper agency sells X copies of a certain newspaper for Rs.A per copy. The cost to the agency of each newspaper is Rs.B . The agency pays a fixed cost for storage, delivery and so on of Rs.100 per Sunday. The newspaper agency wants to calculate the profit obtained on Sundays. Can you please help them out by writing a C program to compute the profit given X, A and B.

Input Format:

Input consists of 3 integers: X, A and B. X is the number of copies sold, A is the cost per copy and B is the cost the agency spends per copy.

Output Format:

Refer Sample Input and Output for exact formatting specifications.

Sample Input and Output:

Input

1000

2

1

Output

900

For example:

Input	Result
1000 2 1	900

Answer: (penalty regime: 0 %)

Each Sunday, a newspaper agency sells X copies of a certain newspaper for Rs.A per copy. The cost to the agency of each newspaper is Rs.B . The agency pays a fixed cost for storage, delivery and so on of Rs.100 per Sunday. The newspaper agency wants to calculate the profit obtained on Sundays. Can you please help them out by writing a C program to compute the profit given X, A and B.

Input Format:

Input consists of 3 integers: X, A and B. X is the number of copies sold, A is the cost per copy and B is the cost the agency spends per copy.

Output Format:

Refer Sample Input and Output for exact formatting specifications.

Sample Input and Output:

Input

1000

2

1

Output

900

For example:

Input	Result
1000 2 1	900

Answer: (penalty regime: 0 %)

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int X,A,B;
    int fixedcost = 100;
    int profit;
    scanf("%d %d %d",&X,&A,&B);
    profit = (X * A) - (X * B) -fixedcost;
    printf("%d\n", profit);
    return 0;
}
```

	Input	Expected	Got	
✓	1000 2 1	900	900	✓

Passed all tests! ✓

Finish review

Baba is very kind to beggars and every day Baba donates half of the amount he has when ever a beggar requests him. The money M left in Baba's hand is passed as the input and the number of beggars B who received the alms are passed as the input. The program must print the money Baba had in the beginning of the day.

Input Format:

The first line denotes the value of M.
The second line denotes the value of B.

Output Format:

The first line denotes the value of money with Baba in the beginning of the day.

Example Input/Output:

Input:

100
2

Output:

400

Explanation:

Baba donated to two beggars. So when he encountered second beggar he had 100*2 = Rs.200 and when he encountered 1st he had 200*2 = Rs.400.

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int M,B,donate,Amount;
    scanf("%d%d",&M,&B);
    donate=M*B;
    Amount=donate*2;
    printf("%d",Amount);
    return 0;
}
```

	Input	Expected	Got	
✓	100 2	400	400	✓

Passed all tests! ✓

The CEO of company ABC Inc wanted to encourage the employees coming on time to the office. So he announced that for every consecutive day an employee comes on time in a week (starting from Monday to Saturday), he will be awarded Rs.200 more than the previous day as "Punctuality Incentive". The incentive I for the starting day (ie on Monday) is passed as the input to the program. The number of days N an employee came on time consecutively starting from Monday is also passed as the input. The program must calculate and print the "Punctuality Incentive" P of the employee.

Input Format:

The first line denotes the value of I.
The second line denotes the value of N.

Output Format:

The first line denotes the value of P.

Example Input/Output:

Input:

500
3

Output:

2100

Explanation:

On Monday the employee receives Rs.500, on Tuesday Rs.700, on Wednesday Rs.900

```c
#include<stdio.h>
int main ()
{
    int I,N;
    int P = 0;
    scanf("%d", &I);
    scanf("%d", &N);
    for(int day = 0;day < N;day++)
    {
        P += I + (200 * day);
    }
    printf("%d\n", P);
    return 0;
}
```

	Input	Expected	Got	
✓	500 3	2100	2100	✓
✓	100 3	900	900	✓

Bajan Lal distributes C chocolates to school N students every Friday. The C chocolates are distributed among N students equally and the remaining chocolates R are given back to Bajan Lal.

As an example if C=100 and N=40, each student receives 2 chocolates and the balance 100-40*2 = 20 is given back.

If C=205 and N=20, then each student receives 10 chocolates and the balance 205-20*10 = 5 is given back.

Help the school to calculate the chocolates to be given back when C and N are passed as input.

Input Format:

The first line denotes C
The second line denotes N

Output Format:

The first line denotes R - the number of chocolates to be given back.

Example Input/Output:

Input:

300
45

Output:

30

```c
#include<stdio.h>
int main()
{
    int C, N,R ;

    scanf("%d",&C);

    scanf("%d",&N);

    R = C % N ;
    printf(" %d\n",R);
    return 0 ;
}
```

	Input	Expected	Got	
✓	300 45	30	30	✓

Passed all tests! ✓

The general format of if statement is

```
if (condition) {
   statement-1;
   statement-2;
   ....
   statement-n;
}
```

The if construct is a **selective statement**, the statements within the block are executed only once when the **condition evaluates to true**, otherwise the control goes to the first statement after the if construct.

If only one statement is presented in the if construct then there is no need to specify the braces {, } i.e., if braces are not specified for the if construct, by default the next immediate statement is the only statement considered for the if construct.

Below code prints the number only when it is **divisible by 3**:

```
#include <stdio.h>
int main()
{
        int num;
        printf("Enter a number : ");
        scanf("%d", &num);
        if (num % 3 == 0)
        {
                printf("Given number %d is divisible by 3", num);
        }
        return 0;
}
```

In the above code, num % 3 == 0 is the **condition**, which verifies whether the **number is divisible by 3**. Only if the condition returns 1 (true) then the control enters in to the **if-block** and executes the statement.

The general format of if statement is

```
if (condition) {
   statement-1;
   statement-2;
   ....
   statement-n;
}
```

The if construct is a **selective statement**, the statements within the block are executed only once when the **condition evaluates to true**, otherwise the control goes to the first statement after the if construct.

If only one statement is presented in the if construct then there is no need to specify the braces { } i.e., if braces are not specified for the if construct, by default the next immediate statement is the only statement considered for the if construct.

Below code prints the number only when it is **divisible by 3**:

```
#include <stdio.h>
int main()
{
        int num;
        printf("Enter a number : ");
        scanf("%d", &num);
        if (num % 3 == 0)
        {
                printf("Given number %d is divisible by 3", num);
        }
        return 0;
}
```

In the above code, num % 3 == 0 is the **condition**, which verifies whether the **number is divisible by 3**. Only if the condition returns 1 (true) then the control enters in to the **if-block** and executes the statement.

Fill in the missing code in the below program to check whether the given number is divisible by **3** or not.

For example:

then the control enters in to the **if-block** and executes the statement.

Fill in the missing code in the below program to check whether the given number is divisible by **3** or not.

For example:

Input	Result
9	Given number 9 is divisible by 3
7	Given number 7 is not divisible by 3

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>

int main()
{
    int num;
    scanf("%d", &num);
    if(num % 3 == 0 )
    {
        printf("Given number %d is divisible by 3\n", num);
    }
    if(num %3 != 0)
    {
        printf("Given number %d is not divisible by 3\n", num);
    }
    return 0;
}
```

```c
int main()
{
    int num;
    scanf("%d", &num);
    if(num % 3 == 0 )
    {
        printf("Given number %d is divisible by 3\n", num);
    }
    if(num %3 != 0)
    {
        printf("Given number %d is not divisible by 3\n", num);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	9	Given number 9 is divisible by 3	Given number 9 is divisible by 3	✓
✓	7	Given number 7 is not divisible by 3	Given number 7 is not divisible by 3	✓

Passed all tests! ✓

Question 1

Correct

Marked out of 1.00

Flag question

The if statement tells a program to execute a certain section of code only if a particular test evaluates to true. if (*expression*) {*statement*}.

Below is a sample code which uses a if statement:

```
int distinction_marks = 75;
if (marks > distinction_marks)
{
        printf("User secured distinction.\n");
}
```

An if statement will execute its block only when condition evaluates to 1 (**true**).

We can also conditionally execute another block when the condition evaluates to 0 (**false**) using the else construct. The else construct must be attached to an if, hence together they are referred to as if-else construct.

The if-else statement provides two different paths of execution depending on the result of the condition.

Below is the general syntax for the if-else statement :

```
if (expression)
{
        statement-1;
}
else
{
        statement-2;
}
```

Below is an example with code:

```
int distinction_marks = 75;
if (marks > distinction_marks)
{
        printf("User secured distinction.\n");
}
else
{
```

```
int distinction_marks = 75;
if (marks > distinction_marks)
{
        printf("User secured distinction.\n");
}
else
{
        printf("User did not secure distinction.\n");
}
```

Fill in the missing code in the below program to check whether the user secured distinction or not.

For example:

Input	Result
76	User secured distinction.
21	User did not secure distinction.

Answer: (penalty regime: 0 %)

Reset answer

```
#include <stdio.h>

int main()
{
    int marks, distinction_marks = 75;
    scanf("%d", &marks);
    if(marks > distinction_marks)
    { // Write the if condition
        printf("User secured distinction.\n");
    }
    else
    { // Write else part
        printf("User did not secure distinction.\n");
    }
}
```

	Input	Expected	Got	
✓	76	User secured distinction.	User secured distinction.	✓
✓	21	User did not secure distinction.	User did not secure distinction.	✓

Passed all tests! ✓

Write code which uses an if-else statement to check whether a given account balance is greater or lesser than the minimum balance.

Use the if-else statement and print "Balance is low" if the balance is less than **1000**, otherwise print "Sufficient balance".

For example, if the user gives the **input** as 1500:

```
1500
```

then the program should **print** the result as:

```
Sufficient balance
```

Similarly, if the input is given as 700 then print

```
Balance is low
```

[**Hint:** Make sure to read the input as a float value.]

For example:

Input	Result
1225	Sufficient balance
999.55	Balance is low

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main ()
{
    int balance;
    scanf("%d",&balance);
    if(balance>=1000)
    {
        printf("Sufficient balance");
    }
    else
    {
        printf("Balance is low");
    }
    return 0;

}
```

	Input	Expected	Got	
✓	1225	Sufficient balance	Sufficient balance	✓
✓	999.55	Balance is low	Balance is low	✓

Passed all tests! ✓

Fill in the missing code in the below program to check whether the student secured first class or not.

Note-1: Read **6** subjects marks, find total and percentage, then print the student secured first class or not.

Note-2: If percentage is greater than or equal to **60** then print student secured first class and the percentage.

For example:

Input	Result
45 67 34 57 68 81	Student did not secure a first class with 58.67%
67 68 65 56 59 69	Student secured a first class with 64.00%

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>

int main()
{
    int maths_marks, computers_marks, physics_marks, chemistry_marks, english_marks, spanish_marks, total
    float percentage;
    // Read marks

    scanf("%d %d %d %d %d %d",&maths_marks,&computers_marks,&physics_marks,&chemistry_marks,&english_mark
    // Calculate total and percentage
    total = maths_marks + computers_marks + physics_marks + chemistry_marks + english_marks + spanish_mar
    percentage = (total / 6.0);
    if(percentage >= 60)
    { // Write the condition
        printf("Student secured a first class with %5.2f%%\n", percentage);
    }
    else
    { // Write the else part
        printf("Student did not secure a first class with %5.2f%%\n", percentage);
    }
    return 0;
}
```

	Input	Expected	Got
✓	45 67 34 57 68 81	Student did not secure a first class with 58.67%	Student did not secure a first class with 5
✓	67 68 65 56 59 69	Student secured a first class with 64.00%	Student secured a first class with 64.00%

Passed all tests! ✓

Write a program which uses an if-else statement to verify and print if the given number is an odd or an even.

For example, if the user gives the **input** as 10:

```
10
```

then the program should **print** the result as:

```
The given number 10 is an even number
```

If the input is given as 35, then the program should print the result as :

```
The given number 35 is an odd number
```

For example:

Input	Result
35	The given number 35 is an odd number
10	The given number 10 is an even number

```c
#include <stdio.h>

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

    if(number%2 == 0)
    { // write if condition to check the given number is even or odd
      // print even or odd
      printf("The given number %d is an even number",number);
    }
    else
    { // print even or odd
       printf("The given number %d is an odd number",number);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	35	The given number 35 is an odd number	The given number 35 is an odd number	✓
✓	10	The given number 10 is an even number	The given number 10 is an even number	✓

Passed all tests! ✓

Write a program which uses an if-else statement to verify if the given character is an alphabet or not.

For example, if the user gives the **input** as W:

```
W
```

then the program should **print** the result as:

```
Given character W is an alphabet
```

If the input us given as 7, then print the result as:

```
Given character 7 is not an alphabet
```

[**Hint:** The ASCII values of alphabets '**A**' to '**Z**' are 65 to 90 and '**a**' to '**z**' are 97 to 122.]

For example:

Input	**Result**
W	Given character W is an alphabet
7	Given character 7 is not an **alphabet**

```c
#include<stdio.h>
int main()
{
    char ch;

    scanf("%c",&ch);
    if((ch >= 'A' && ch <= 'Z')||(ch >= 'a' && ch <= 'z'))
    {
        printf("Given character %c is an alphabet\n", ch);
    } else
    {
        printf("Given character %c is not an alphabet\n", ch);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	W	Given character W is an alphabet	Given character W is an alphabet	✓
✓	7	Given character 7 is not an alphabet	Given character 7 is not an alphabet	✓

Passed all tests! ✓

When an if-else construct appear as a statement within another if-block or a else-block, it is referred to as nesting of if-else construct.

Below is an example of a **nested if-else** construct:

```
if (expression_1)
{
        if (expression_2)
        {
                if (expression_3)
                {
                        statement_1;
                }
                else
                {
                        statement_2;
                }
        }
        else
        {
                statement_3;
        }
}
```

In the above syntax, the **statement_2** will be executed only when the conditions in expression_1, expression_2 and expression_3 evaluates to 1 (true).

Fill in the missing code in the below program to find the **largest** of three numbers using nested if-else.

For example:

Input	Result
23 56 77	77 is greater than 23 and 56

Answer: (penalty regime: 0 %)

Reset answer

Reset answer

```c
#include <stdio.h>

int main()
{
    int a, b, c;

    scanf("%d %d %d", &a, &b, &c);
    if(a > b && a >c)
    {
        printf("%d is greater than %d and %d\n",a,b,c);
    }
    else if(b > a && b > c)
    {
        printf("%d is greater than %d and %d\n",b,a,c);
    }
    else if(c > a && c > b)
    {
        printf("%d is greater than %d and %d\n",c,a,b);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	23 56 77	77 is greater than 23 and 56	77 is greater than 23 and 56	✓

Passed all tests! ✓

Question 1

Correct

Marked out of 1.00

Flag question

The if-else-if construct extends the if-else construct by allowing to chain multiple if constructs as shown below:

```
if (expression_1)
{
        statement_1;
}
else if (expression_2)
{
        statement_2;
}
else if (expression_3)
{
        statement_3;
}
else if (expression_4)
{
        statement_4;
}
else
{
        statement_5;
}
```

As shown in the above syntax, multiple if constructs can be chained to any length. The else construct which appears at the end is optional, and if it is to be included it has to be only at the end.

The if-else-if construct is used whenever we have multiple mutually exclusive if conditions which work on the same input.

In a if-else-if construct the conditions are evaluated from top to bottom. Whenever a condition evaluates to **true** (1), the control enters into that if-block and after that the control comes out of the complete if-else-if construct ignoring all the remaining if and else constructs that may exist below the currently satisfied if-block.

For example, if the condition in the expression_2 is the first condition to evaluate to **true** after executing statement_2 the control comes out of the complete if-else-if construct.

The below program reads a character from the console and should print if the given character is an alphabet or a digit. Do not remove the existing code, add the missing lines of code which employs the if-else-if statement to produce appropriate output.

For example:

Input	Result
A	Given character A is an alphabet
8	Given character 8 is a digit
%	Given character % is neither an alphabet nor a digit

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>

int main()
{
    char ch;
    ch = getchar();
    if((ch >= 'A' && ch <= 'Z') ||(ch >= 'a' && ch <= 'z'))
    printf("Given character %c is an alphabet\n", ch);
    else if(ch >= '0' && ch <= '9')
    printf("Given character %c is a digit\n", ch);
    else
    printf("Given character %c is neither an alphabet nor a digit\n", ch);
    return 0;
}
```

	Input	Expected	Got
✓	A	Given character A is an alphabet	Given character A is an alphabet
✓	8	Given character 8 is a digit	Given character 8 is a digit
✓	%	Given character % is neither an alphabet nor a digit	Given character % is neither an alphabet nor a digit

Passed all tests! ✓

Question **2**

Correct

Marked out of 1.00

Flag question

The following code uses if-else statement to check whether the given integer number is a valid **leap year** or not.

Use if-else statement and print "__ is a leap year":

- if a year is divisible by **4** and should not be divisible by **100**.
- If a year is divisible by **400**.

Otherwise, print "__ is not a leap year".

Fill in the missing code in the below program to check whether the given year is a **leap year** or not..

For example:

Input	Result
1900	1900 is not a leap year

Reset answer

```c
#include <stdio.h>

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

    if(year % 4 == year/2)
    {
        printf("%d is a leap year\n", year);
    }
    else if(year %4 !=0)
    {
        printf("%d is not a leap year\n", year);
    }
    else if(year % 400 ==0)
    {
        printf("%d is a leap year\n", year);
    }
    else
    {
        printf("%d is not a leap year\n", year);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	1900	1900 is not a leap year	1900 is not a leap year	✓
✓	2000	2000 is a leap year	2000 is a leap year	✓

Passed all tests! ✓

Fill in the missing code in the below program to read an **integer value** for a variable age and use if-else statement to check the age and print appropriate ticket price.

If **age** is lessthan or equal to **infant_age** (3 years) or greaterthan or equal to **centenarian_age** (100 years) then print **Ticket Price: 0**.

Otherwise, If **age** is lessthan or equal to **child_age** (13 years) or greaterthan or equal to **senior_citizen_age** (60 years) then print **Ticket Price: 5**.

Otherwise, print **Ticket Price: 10**.

For example:

Input	Result
34	`Ticket Price: 10`
2	`Ticket Price: 0`
101	`Ticket Price: 0`
72	`Ticket Price: 5`

```c
#include <stdio.h>

int main()
{
    int age, infant_age = 3, child_age = 13, senior_citizen_age = 60, centenarian_age = 100;
    scanf("%d", &age);
    if(age <= infant_age || age >= centenarian_age)
    { // if condition
        printf("Ticket Price: 0\n");
    }
    else if(age <= child_age ||  age >= senior_citizen_age)
    { // else if condition
        printf("Ticket Price: 5\n");
    }
    else
    { // else
        printf("Ticket Price: 10\n");
    }
    return 0;
}

```

	Input	Expected	Got	
✓	34	Ticket Price: 10	Ticket Price: 10	✓
✓	2	Ticket Price: 0	Ticket Price: 0	✓
✓	101	Ticket Price: 0	Ticket Price: 0	✓
✓	72	Ticket Price: 5	Ticket Price: 5	✓

Passed all tests! ✓

Question **4**

Correct

Marked out of 1.00

Flag question

See the below code which uses a if-else-if statement for calculating **AM** or **PM** for a given **hour**.

In the **main()** function read an integer value between **0** and **23** for the variable hour and use if-else-if statement to display **AM** or **PM**.

Fill in the if condition to check if the given hour is between **0** and **11** (both inclusive) for **AM**. Fill in the else if condition to check if the given hour is between **12** and **23** (both inclusive) for **PM**.

For example:

Input	Result
9	AM
22	PM
24	Invalid Hour!!

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main()
{
    int hour;
    scanf("%d", &hour);
    if ( hour >= 0 && hour <=11 )
    { //fill the condition for AM here
        printf("AM\n");
    }
    else if (hour >= 12 && hour <= 23     )
    { //fill the condition for PM here
        printf("PM\n");
    }
    else
    {
        printf("Invalid Hour!!\n");
    }
    return 0;
}
```

	Input	Expected	Got	
✓	9	AM	AM	✓
✓	22	PM	PM	✓
✓	24	Invalid Hour!!	Invalid Hour!!	✓

Passed all tests! ✓

A switch statement is used to change the control flow of a program execution through multiple paths depending on an expression's value.

The below code demonstrates how to use a switch-case construct to print the corresponding English words for the digits (**1** to **9**) read from the standard input.

One way is to write a long nested if-else-if for the **10** numbers or the other way is to use a switch-case statement.

See and retype the below code which demonstrates the usage of switch statement to print the English word of the given number between **1** to **9**.

```
#include <stdio.h>
int main()
{
   int value;
   scanf("%d", &value);
   switch (value)
   {
   case 1:
      printf("One");
      break;
   case 2:
      printf("Two");
      break;
      case 3:
      printf("Three");
      break;
      case 4:
      printf("Four");
         break;
      case 5:
      printf("Five");
      break;
      case 6:
      printf("Six");
      break;
      case 7:
      printf("Seven");
```

```
        case 4:
        printf("Four");
           break;
        case 5:
        printf("Five");
        break;
        case 6:
        printf("Six");
        break;
        case 7:
        printf("Seven");
           break;
        case 8:
        printf("Eight");
           break;
        case 9:
           printf("Nine");
           break;
        case 10:
           printf("Ten");
           break;
        default:
           printf("Number %d is not in the range 1 to 10", value);
    }
    return 0;
}
```

For example:

Input	Result
2	Two
9	Nine
15	Number 15 is not in the range 1 to 10

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int value;
    scanf("%d",&value);
    switch(value)
    {
        case 1:
        printf("One");
        break;
        case 2:
        printf("Two");
        break;
        case 3:
        printf("Three");
        break;
        case 4:
        printf("Four");
        break;
        case 5:
        printf("Five");
        break;
        case 6:
        printf("Six");
        break;
        case 7:
        printf("Seven");
        break;
        case 8:
        printf("Eight");
        break;
        case 9:


        printf("Nine");
        break;
        case 10:
        printf("Ten");
        break;
        default:
        printf("Number %d is not in the range 1 to 10",value);

    }
}
```

	Input	Expected	Got	
✓	2	Two	Two	✓
✓	9	Nine	Nine	✓
✓	15	Number 15 is not in the range 1 to 10	Number 15 is not in the range 1 to 10	✓

Passed all tests! ✓

Assume that the weekdays are provided with the below numbers:

```
Sunday ⟹ 0
Monday ⟹ 1
Tuesday ⟹ 2
Wednesday ⟹ 3
Thursday ⟹ 4
Friday ⟹ 5
Saturday ⟹ 6
```

Write a program to read the **weekday number** from the standard input and print the **weekday name** using switch-case.

For example, if the user gives the **input** as 1:

```
1
```

then the program should **print** the result as:

```
Monday
```

Note: If the given input number is not in the range i.e., other than **0** to **6**, the output should be as given below:

```
Invalid weekday number
```

For example:

Input	Result
6	Saturday
0	Sunday
7	Invalid weekday number

```c
#include<stdio.h>
int main()
{
    int weekday;
    scanf("%d",&weekday);
    switch(weekday)
    {
        case 0:
        printf("Sunday\n");
        break;
        case 1:
        printf("Monday\n");
        break;
        case 2:
        printf("Tueday\n");
        break;
        case 3:
        printf("Wednesday\n");
        break;
        case 4:
        printf("Thursday\n");
        break;
        case 5:
        printf("Friday\n");
        break;
        case 6:
        printf("Saturday\n");
        break;
        default:
        printf("Invalid weekday number\n");
    }
    return 0;
}
```

	Input	Expected	Got	
✓	6	Saturday	Saturday	✓
✓	0	Sunday	Sunday	✓
✓	7	Invalid weekday number	Invalid weekday number	✓

Question **1**

Correct

Marked out of 1.00

Flag question

Most of the programming languages provide a special construct/statement using which we can repeatedly execute one or more statement as long as a condition is **true**. In C, we have while, do-while and for as the three main looping constructs or statements.

Below is a general syntax for using a while statement:

```
while (condition)
{
        statement_1;
        statement_2;
        ....
}
```

The block of code inside the opening and closing brace which follows the while-statement is called the **while-loop** body.

A while statement is used to execute some code repeatedly as long as a condition evaluates to true.

The condition is an expression which should always evaluate to either true or false.

- If it evaluates to true, the body containing one or more code statements is executed.
- If the expression evaluates to false, the control skips executing the **while-loop** body.

The while-loop construct is also referred to as an entry controlled loop. Meaning, first the condition is evaluated and only if the condition evaluates to true the body of the loop is executed. After executing the body the control is automatically transferred back to the condition and the process continues until the condition evaluates to false.

See and retype the below code which uses a while-loop to read multiple numbers from standard input and prints their sum when the **sum** exceeds 100.

```
#include <stdio.h>

int main()
{
   int total = 0;
   while (total <= 100)
```

```
    {
    int num;
    scanf("%d", &num);
       total += num;
    }
    printf("The total of given numbers is : %d", total);
    return 0;
}
```

For example:

Input	Result
34 62 24	The total of given numbers is : 120

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int total=0;

    while(total<=100)
    {
        int num;
        scanf("%d",&num);
        total+=num;
    }
    printf("The total of given numbers is : %d",total);
    return 0;
}
```

	Input	Expected	Got	
✓	34 62 24	The total of given numbers is : 120	The total of given numbers is : 120	✓

Passed all tests! ✓

The below sample code should print Ganga by number of times, where as the input is read by the programmer using **scanf()**.

Fill in the missing code so that it produces the desired output.

For example:

Input	Result
3	Ganga Ganga Ganga

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>
int main()
{
    int i = 0, n;
    scanf("%d",&n); //Fill the missing code in scanf() function
    while (i<n)
    { // complete the condition here
        printf("Ganga\n"); // Write the text to be printed here
        i++; // Complete the statement
    }
    return 0;
}
```

	Input	Expected	Got	
✓	3	Ganga Ganga Ganga	Ganga Ganga Ganga	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of 1.00

Flag question

Write a **C** program to print first n **natural numbers**.

For example, if the user gives the **input** as :

3

then the program should **print** the result as:

```
The natural numbers from 1 - 3 : 1 2 3
```

For example:

Input	Result
3	The natural numbers from 1 - 3 : 1 2 3
9	The natural numbers from 1 - 9 : 1 2 3 4 5 6 7 8 9

```c
#include<stdio.h>
int main()
{
    int n,i=1;
    scanf("%d", &n);
    printf("The natural numbers from 1 - %d : ",n);
    while(i <=n )
    {
        printf("%d ", i);
        i++;
    }
    printf("\n");
    return 0;
}
```

	Input	Expected	Got	
✓	3	The natural numbers from 1 - 3 : 1 2 3	The natural numbers from 1 - 3 : 1 2 3	✓
✓	9	The natural numbers from 1 - 9 : 1 2 3 4 5 6 7 8 9	The natural numbers from 1 - 9 : 1 2 3 4 5 6 7 8 9	✓

Passed all tests! ✓

The below sample code should find the sum of **even numbers** between any two numbers.

[**Hint:** The numbers should be read by using scanf()].

Fill in the missing code so that it produces the desired output.

For example:

Input	Result
3 6	The sum of even integers between the given limits = 10

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>

int main()
{
    int start, end, sum = 0;
    scanf("%d",&start);
    scanf("%d",&end);
    while(start<=end)
    {
    if (start % 2 == 0)
    {
        sum+=start;
    }

    start++;



    }
    printf("The sum of even integers between the given limits = %d\n",sum);
    return 0;
}
```

	Input	Expected	Got
✓	3 6	The sum of even integers between the given limits = 10	The sum of even integers between the given limits

Passed all tests! ✓

Question **5**

Correct

Marked out of 1.00

Flag question

Fill in the missing code in the below program to read an **integer number** and find the reverse of the given number.

For example if the input is 1234, then the output will be 4321.

Hints

The logic of reversing of any number is pretty simple if you know how to find last digit of any number. Initially the variable reverse contains zero(0), the process of reversing involves four basic steps:

- Multiply the reverse variable by 10.
- Find the last digit of the given number by applying % 10.
- Add the last digit just found to reverse.
- Divide the original number by 10 to eliminate the last digit, which is not needed anymore.

Repeat the above four steps till the original number becomes 0 and finally we will be left with the reversed number in reverse variable.

For example:

Fill in the missing code in the below program to read an **integer number** and find the reverse of the given number.

For example if the input is 1234, then the output will be 4321.

Hints

The logic of reversing of any number is pretty simple if you know how to find last digit of any number. Initially the variable reverse contains zero(0), the process of reversing involves four basic steps:

- Multiply the reverse variable by 10.
- Find the last digit of the given number by applying % 10.
- Add the last digit just found to reverse.
- Divide the original number by 10 to eliminate the last digit, which is not needed anymore.

Repeat the above four steps till the original number becomes 0 and finally we will be left with the reversed number in reverse variable.

For example:

Input	Result
1234	`The reverse number of a given number = 4321`
765	`The reverse number of a given number = 567`

```c
#include <stdio.h>

int main()
{
    int n,  reverse = 0,remainder,original;
    scanf("%d", &n);
    original=n;
    while (n!=0)
    { // Write the condition
    remainder=n%10;// Fill the correct code
    reverse =reverse*10+remainder; // Fill the correct code
    n/=10; // Fill the correct code
    }
    if(original%10==0)
    {
        printf("The rverse number of a given number = %d",reverse);
        while(original%10==0)
        {
            printf("0");
            original/=10;
        }
    }
    else
    {
        printf("The reverse number of a given number = %d",reverse);
    }

    return 0;
}
```

	Input	Expected	Got	
✓	1234	The reverse number of a given number = 4321	The reverse number of a given number = 4321	✓
✓	765	The reverse number of a given number = 567	The reverse number of a given number = 567	✓

Passed all tests! ✓

Fill in the missing code in the below sample program which finds the factorial of a given number.

Factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n.
For example, 5! = 5 * 4 * 3 * 2 * 1 = 120.

The below sample code computes the factorial of a given non-zero integer.

The main() function declares an integer variable factorial and initializes it to 1, which it will use to store the computed factorial value.

It uses a **while-loop** to iterate from 2 to n multiplying the loop counter in each iteration with the factorial and storing the product again in factorial.

For example:

Input	Result
2	Factorial of given number 2 = 2
4	Factorial of given number 4 = 24

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>

int main()
{
    int i, n, factorial = 1;
    scanf("%d", &n);
    i = 2;
    while (i<=n )
    { // Write the condition
    factorial =factorial*i  ; // Fill the correct code
    i++;
    }
    printf("Factorial of given number %d = %d\n", n, factorial);
    return 0;
}
```

	Input	Expected	Got	
✓	2	Factorial of given number 2 = 2	Factorial of given number 2 = 2	✓
✓	4	Factorial of given number 4 = 24	Factorial of given number 4 = 24	✓

Passed all tests! ✓

Below partial code is to verify if the given number is a prime number or not.

A prime number is a positive integer greater than 1, which is not divisible by any other number other than 1 and itself. Examples of a few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, etc.

Fill in the missing code so that it produces the desired output.

For example:

Input	Result
7	`The given number 7 is a prime number`
119	`The given number 119 is not a prime number`

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>

int main()
{
    int n, i =2 , count =0 ; // initialize i and count with appropriate values
    scanf("%d", &n);
    while (i<=n/2 )
    { // complete the condition to iterate the loop
    if (n%i==0 )
        { // complete the condition to check the remainder is 0 or not
        count++;
    }
    i++;
    }
    if (count ==0  )
    { // complete the condition to check the count
        printf("The given number %d is a prime number\n", n);
    }
    else
    {
    printf("The given number %d is not a prime number\n", n);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	7	The given number 7 is a prime number	The given number 7 is a prime number	✓
✓	119	The given number 119 is not a prime number	The given number 119 is not a prime number	✓

Passed all tests! ✓

Below partial code is to verify if the given number is an armstrong number or not.

An armstrong number is a number that is the sum of its own digits raised to the power of number of digits that make up the original number.

For example, if the given number is 153, the total number of digits are 3, and the sum of cubes of each digit ($1^3 + 5^3 + 3^3$) is equal to the same number 153. Such a number is known as an armstrong number.

Let us take another example, if the given number is 9474, the total number of digits are 4, and the sum of the power of 4 of each digit ($9^4 + 4^4 + 7^4 + 4^4$) is equal to the same number 9474. Such a number is known as an armstrong number.

Similarly,
$9 = 9^1 = 9$
$371 = 3^3 + 7^3 + 1^3 = 27 + 343 + 1 = 371$
$3\ 8208 = 8^4 + 2^4 + 0^4 + 8^4 = 4096 + 16 + 0 + 4096 = 8208$

Fill in the missing code so that it produces the desired output.

For example:

Input	Result
777	The given number 777 is not an armstrong number
9	The given number 9 is an armstrong number

```c
#include <stdio.h>

int main()
{
    int number, temp, remainder, i, power, digits = 0, sum = 0;
    scanf("%d", &number);
    temp = number;
    while (temp != 0)
    { // complete the condition to iterate the loop
    digits++ ; // increment the digits
    temp /= 10 ; //calculate the temp value
    }
    temp = number;
    while (temp != 0)
    { // complete the condition to iterate the loop
    remainder = temp % 10;
    i = 1;
    power = 1;
    while (i <= digits)
        { // find the powers of each digit
        power *= remainder; // calculate power value
            i++; // increment i value
        }
        sum += power; // calculate sum value
        number /= 10; // calculate number value
    }
    if (sum == number)
    { // write the condition
        printf("The given number %d is an armstrong number\n", number);
    }
    else
    {
    printf("The given number %d is not an armstrong number\n", number);
    }
    return 0;
}
```

	Input	Expected	Got	
✗	777	The given number 777 is not an armstrong number	***Time limit exceeded***	✗

Testing was aborted due to error.

Your code must pass all tests to earn any marks. Try again.

Show differences

Question **1**

Correct

Marked out of 1.00

Flag question

A for-loop is used to iterate over a range of values using a loop counter, which is a variable taking a range of values in some orderly sequence (e.g., starting at 0 and ending at 10 in increments of 1).

The value stored in a loop counter is changed with each iteration of the loop, providing a unique value for each individual iteration. The loop counter is used to decide when to terminate the loop.

A for-loop construct can be termed as an entry controlled loop.
Below is the syntax of a **for-loop** :

```
for (initialization; condition; update)
{
    statement(s);
}
```

1. The initialization expression initializes the loop counter; it is executed **once** at the start of the loop.
2. The loop continues to execute as long as the condition expression evaluates to true.
3. The update expression is executed after each iteration through the loop, to **increment**, **decrement** or **change** the **loop counter**.

Example with code :

```
int i;
for (i = 0; i < 10; i++)
{
   printf("%d\n",i);
}
```

1. Above **for-loop** statement initializes an integer variable i (which is the **loop counter**) as part of the initialization expression.
2. In the update section, it increments the variable i by **1** using the post-increment expression i++.
3. The expression in condition is i < 10. The for-loop keeps on executing the code inside the loop body as long as this condition evaluates to true. And the loop terminates when the condition evaluates to false.
4. It is a good practice to always keep the loop body (which contains the code to be executed) within an opening-brace { and a closing-brace }.

Note : No ; at the end of the for statement.

Complete the below code to check your understanding of the for-loop syntax. The completed code should print numbers from 10 to 20, one

Note : No ; at the end of the for statement.

Complete the below code to check your understanding of the for-loop syntax. The completed code should print numbers from 10 to 20, one per line.

For example:

Result
10 11 12 13 14 15 16 17 18 19 20

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 10;i <= 20; i++)
    {
        printf("%d\n", i);
    }
    return 0;
}
```

	Expected	Got	
✓	10	10	✓
	11	11	
	12	12	
	13	13	
	14	14	
	15	15	
	16	16	
	17	17	
	18	18	
	19	19	
	20	20	

Passed all tests! ✓

Fill in the missing code in the below program to calculate the value of a^n, given two positive non-zero integers a and n.

The code in the main() function reads two integers from standard input and stores them in the variables a and n.

It uses a for-loop to multiply a with itself n number of times.

Variable a_power_n is used to store the computed value of a^n.

After the execution of for-loop is completed, the final value of a_power_n is printed to the standard output.

For example:

Input	Result
2 3	8

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>
int main()
{
    int i, a, n, a_power_n=1;
    scanf("%d %d", &a, &n);
    for ( i = 1 ;i <=n ;i++ )
    { // Write the initialization, condition and increment part
        a_power_n =a_power_n*a   ; // Calculate the value
    }
    printf(" %d\n", a_power_n);
    return 0;
}
```

	Input	Expected	Got	
✓	2 3	8	8	✓

Passed all tests! ✓

Write a program to find **sum** and **mean** of **n** numbers.

Constraints:

- $1 <= n <= 10^6$
- $10^{-3} <=$ elements $<= 10^3$
- Result of mean should print upto **2 decimal places**.

Sample test case:

4----------> First line of input is the value on n.

3 5 7 8----------> Second line of input is n space separated integer values.

Sum: 23----------->Third line prints the Sum as required.

Mean: 5.75----------->Fourth line prints the Mean as required.

Instruction: To run your custom test cases strictly map your input and output layout with the visible test cases.

For example:

Input	Result
4 3 5 7 8	Sum: 23 Mean: 5.75

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int i,a,sum=0;
    float n,mean;
    scanf("%f",&n);
    for(i = 0;i<n;i++)
    {
        scanf("%d",&a);
        sum = sum+a;
    }
    printf("Sum: %d\n",sum);
    mean=sum/n;
    printf("Mean: %.2f",mean);
    return 0;
}
```

	Input	Expected	Got	
✓	4 3 5 7 8	Sum: 23 Mean: 5.75	Sum: 23 Mean: 5.75	✓

Passed all tests! ✓

Fill in the missing code in the below program to print the Fibonacci series i.e., 0 1 1 2 3 5 8 13 21....., up to the limit.

The code in the main() function reads one integer variable n. It uses a for loop to iterate from 0 to n and print the series.

By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

For example:

Input	Result
25	The Fibonacci series is : 0 1 1 2 3 5 8 13 21

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>
int main()
{
    int fib1 = 0, fib2 = 1, fib3, n;
    scanf("%d", &n);
    printf("The Fibonacci series is : %d %d", fib1, fib2);
    for (fib3 = (fib1+fib2)  ; fib3<=n  ; fib3+=fib1 )
    { // Write the initialization, condition and increment part
        printf(" %d", fib3);
        fib1 =fib2; ; // Assign a value
        fib2 =fib3 ; // Assign a value
    }
    return 0;
}
```

	Input	Expected	Got	
✓	25	The Fibonacci series is : 0 1 1 2 3 5 8 13 21	The Fibonacci series is : 0 1 1 2 3 5 8 13 21	✓

Passed all tests! ✓

Write a program that will print all the **English alphabets** from A to Z, each in a new line.

Hints

1. The code in the main() function can use a for loop to iterate over the characters 'A' to 'Z'.
2. Note that char data type is a numeric type and can be used in a for loop as a loop counter.
3. You can declare and initialize a loop counter char i and initialize it to 'A' (eg: char i = 'A';). The condition can similarly be i <= 'Z'; and the update statement can be i++.
4. You can then print i directly which is of type char, using the **printf()** function with a newline character (\n).

For example:

Result
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    for(char i='A';i<='Z';i++)
    {
        printf("%c\n",i);
    }
    return 0;
}
```

	Expected	Got	
✓	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	✓

Passed all tests! ✓

Write a program to read **n** numbers from the user and then count number of "**Odd**" and "**Even**" numbers.

Constraints:

- $1 <= n <= 10^6$
- $10^{-3} <= \text{elements} <= 10^3$

Sample test case:

3---------> First line of input is n i.e. 3.

5 6 7-------> Second line of input is n space separated integer values/elements.

Even: 1-----> Third line prints the output (the count of even elements).

Odd: 2------->Fourth line prints the output (the count of odd elements).

Note: Do use the **printf()** function with a **newline** character (**\n**) to print your results on newline.

Instruction: To run your custom test cases strictly map your input and output layout with the visible test cases.

For example:

Input	Result
3 5 6 7	Even: 1 Odd: 2

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int n,values,even = 0,odd = 0;
    scanf("%d\n",&n);
    for(int i = 1;i <=n;i ++)
    {
        scanf("%d",&values);
        if(values % 2 == 0)
        {
            even++;
        }
        else
        {
            odd++;
        }
    }
    printf("Even: %d\n",even);
    printf("Odd: %d",odd);
    return 0;
}
```

	Input	Expected	Got	
✓	3 5 6 7	Even: 1 Odd: 2	Even: 1 Odd: 2	✓

Passed all tests! ✓

Fill in the missing code in the below program to verify whether the given number is perfect, abundant or deficient.

Fill in the missing code in the below program to verify whether the given number is perfect, abundant or deficient.

A number is said to be perfect if it equals the sum of its proper divisors. For example, **6** and **28** can be called **perfect numbers** as : **6 = 1 + 2 + 3** and **28 = 1 + 2 + 4 + 7 + 14**.

Alternatively, if the sum of a number's proper divisors **exceeds** the number itself, it is said to be abundant, while if the sum of a number's proper divisors is **less-than** the number itself, it is said to be deficient.

For example:

Input	Result
6	`The given number 6 is a perfect number`
10	`The given number 10 is a deficient number`
12	`The given number 12 is an abundant number`

Answer: (penalty regime: 0 %)

Reset answer

Reset answer

```c
#include <stdio.h>
int main()
{
    int n, i, sum = 0;
    scanf("%d", &n);
    for (i = 1; i < n ; i++ )
    { //Write the initialization, condition and increment part
        if (n % i == 0 )
        { // Fill the condition
            sum = sum + i;
        }
    }
    if (sum == n )
    { // Fill the condition
        printf("The given number %d is a perfect number", n);
    }
    else if (sum < n )
    { // Fill the condition
        printf("The given number %d is a deficient number", n);
    }
    else
    {
        printf("The given number %d is an abundant number", n);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	6	The given number 6 is a perfect number	The given number 6 is a perfect number	✓
✓	10	The given number 10 is a deficient number	The given number 10 is a deficient number	✓
✓	12	The given number 12 is an abundant number	The given number 12 is an abundant number	✓

Passed all tests! ✓

Fill in the missing code in the below program to check whether the given number is a strong number or not.

A number is called strong number if sum of the **factorials** of its digit is equal to number itself. For example: **145** is considered a strong number since **1! + 4! + 5! = 1 + 24 + 120 = 145**.

The code in the below main() function reads a number from standard input and performs the verification for a strong number by extracting the individual digits and calculating their factorials.

For example:

Input	Result
145	The given number 145 is a strong number
123	The given number 123 is not a strong number

Answer: (penalty regime: 0 %)

Reset answer

```c
#include <stdio.h>
int main()
{
    int rem, n, i, sum = 0, temp, fact = 1;
    scanf("%d", &n);
    for (temp = n;n>0    ; n = n / 10)
    { // Write the condition part
        rem = n%10 ; // Calculate remainder value
        fact = 1;
        for (i = 1  ; i <= rem ; i++ )
        { // Write the initialization, condition and increment part
            fact = fact * i;
        }
        sum = sum + fact;
    }
    if (sum == temp  )
    { // Fill the condition
        printf("The given number %d is a strong number\n", temp);
    }
    else
    {
        printf("The given number %d is not a strong number\n", temp);
    }
    return 0;
}
```

	Input	Expected	Got	
✓	145	The given number 145 is a strong number	The given number 145 is a strong number	✓
✓	123	The given number 123 is not a strong number	The given number 123 is not a strong number	✓

Passed all tests! ✓