EXERCISE 5

Content:

1. C Loops
2. C do-while loop
3. C while loop
4. C for loop

I. C Loops

The looping can be defined as repeating the same process multiple times until a specific condition satisfies. There are three types of loops used in the C language.

Why use loops in C language?

The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times. For example, if we need to print the first 10 natural numbers then, instead of using the printf statement 10 times, we can print inside a loop which runs up to 10 iterations.

Advantage of loops in C

1) It provides code reusability.

2) Using loops, we do not need to write the same code again and again.

3) Using loops, we can traverse over the elements of data structures (array or linked lists).

Types of C Loops

There are three types of loops in C language that is given below:

1. do while
2. while
3. for

II. C do-while loop

The do while loop is a post tested loop. Using the do-while loop, we can repeat the execution of several parts of the statements. The do-while loop is mainly used in the case where we need to execute the loop at least once. The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end user.

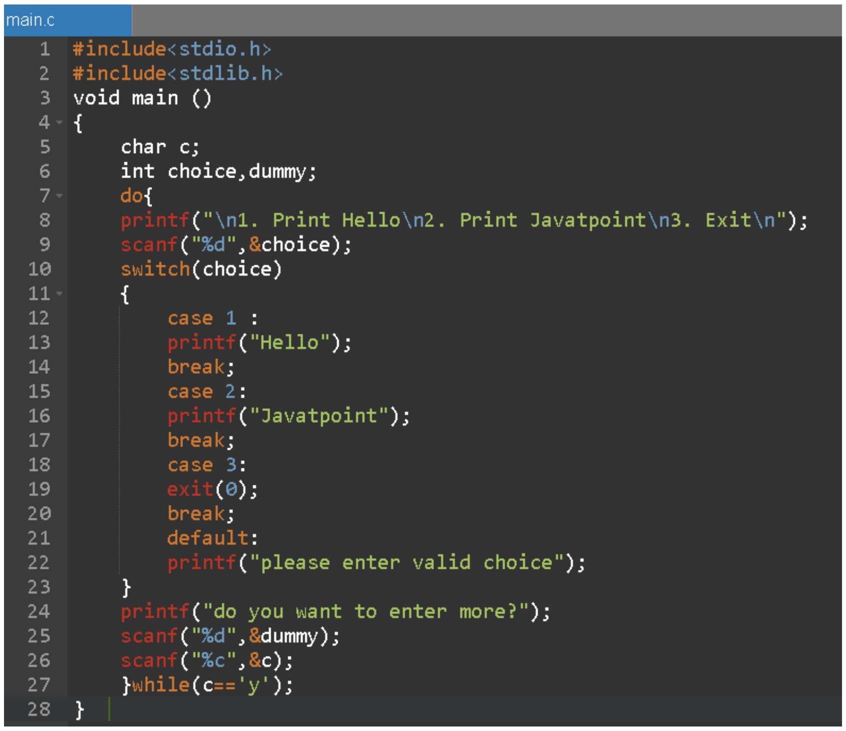
do while loop syntax

The syntax of the C language do-while loop is given below:

```
do{
//code to be executed
}while(condition);
```

IMPORTANT: Now we will change the online compiler CodeChef with this one: https://www.onlinegdb.com/online_c_compiler

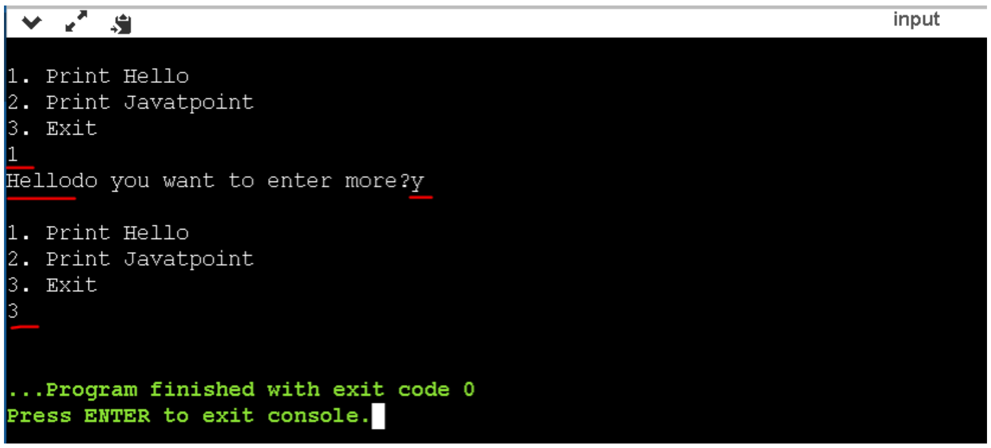
LAB: Type the following code and press RUN button:



main.c

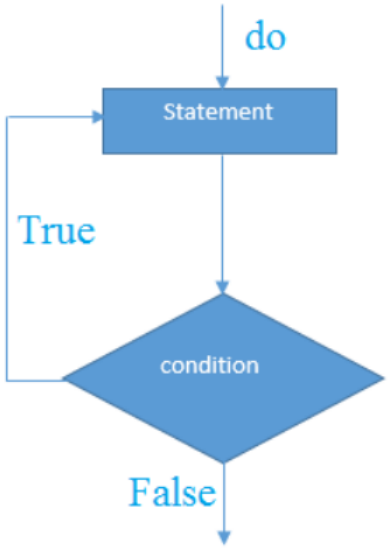
```c
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    char c;
    int choice,dummy;
    do{
    printf("\n1. Print Hello\n2. Print Javatpoint\n3. Exit\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1 :
        printf("Hello");
        break;
        case 2:
        printf("Javatpoint");
        break;
        case 3:
        exit(0);
        break;
        default:
        printf("please enter valid choice");
    }
    printf("do you want to enter more?");
    scanf("%d",&dummy);
    scanf("%c",&c);
    }while(c=='y');
}
```

Result/Output:

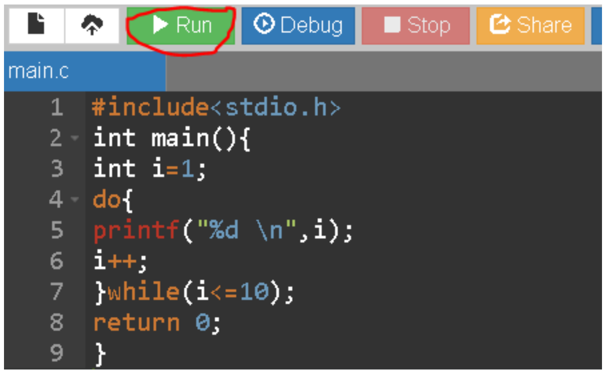


```

1. Print Hello
2. Print Javatpoint
3. Exit
1
Hellodo you want to enter more?y

1. Print Hello
2. Print Javatpoint
3. Exit
3


...Program finished with exit code 0
Press ENTER to exit console.
```

Flowchart of do while loop:

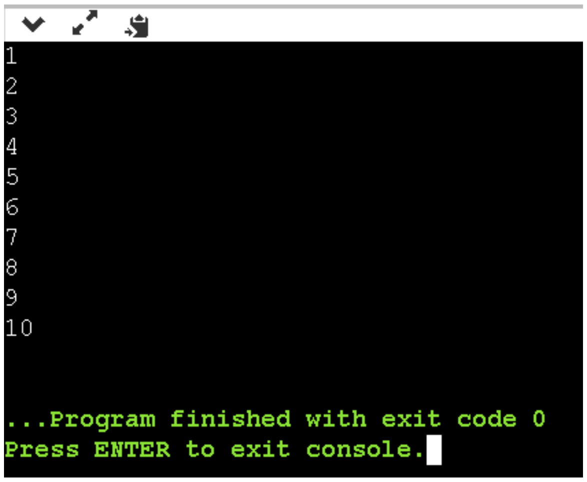


LAB: Simple program of do while loop where we are printing the table of 1.



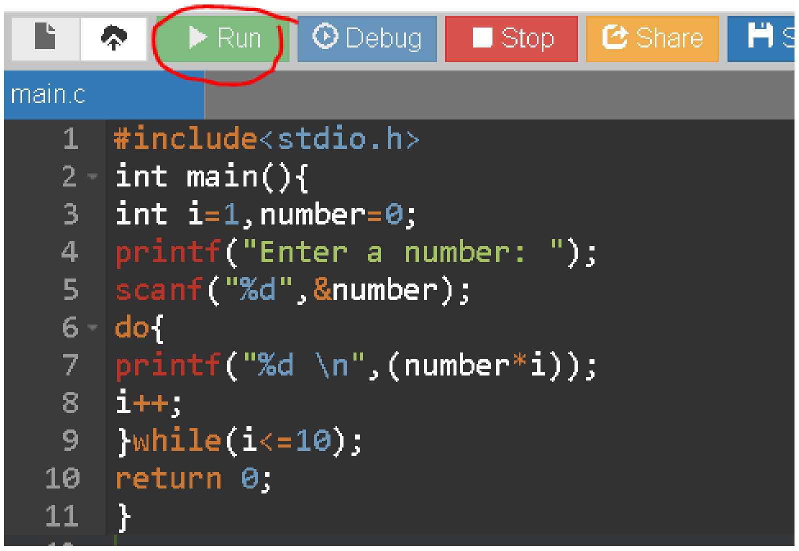
```c
#include<stdio.h>
int main(){
int i=1;
do{
printf("%d \n",i);
i++;
}while(i<=10);
return 0;
}
```

Result/Output:



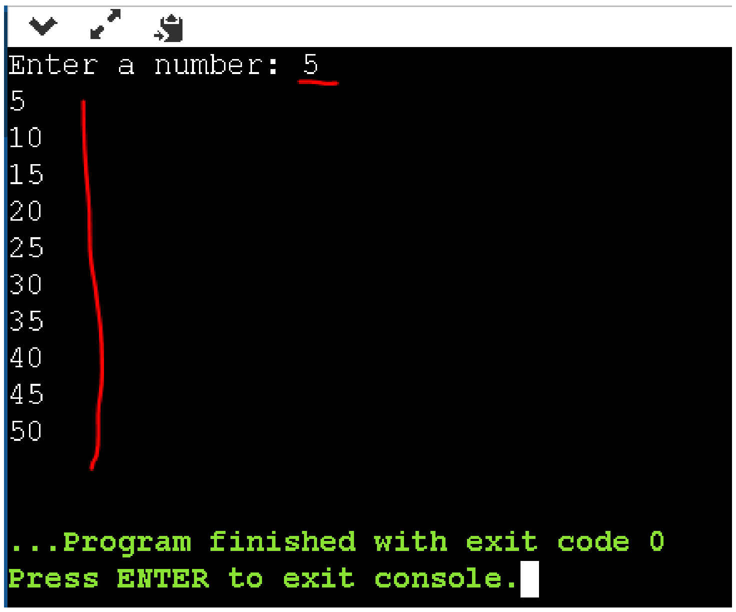
```
1
2
3
4
5
6
7
8
9
10


...Program finished with exit code 0
Press ENTER to exit console.
```

LAB: Program to print table for the given number using do while loop



```
#include<stdio.h>
int main(){
int i=1,number=0;
printf("Enter a number: ");
scanf("%d",&number);
do{
printf("%d \n",(number*i));
i++;
}while(i<=10);
return 0;
}
```

Result/Output:



```
Enter a number: 5
5
10
15
20
25
30
35
40
45
50


...Program finished with exit code 0
Press ENTER to exit console.
```

Infinitive do while loop

The do-while loop will run infinite times if we pass any non-zero value as the conditional expression.

```
do{
//statement
}while(1);
```

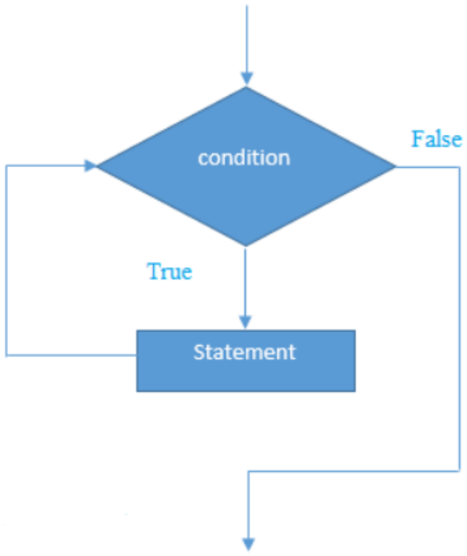
III. C while loop

While loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given boolean condition. It can be viewed as a repeating if statement. The while loop is mostly used in the case where the number of iterations is not known in advance.

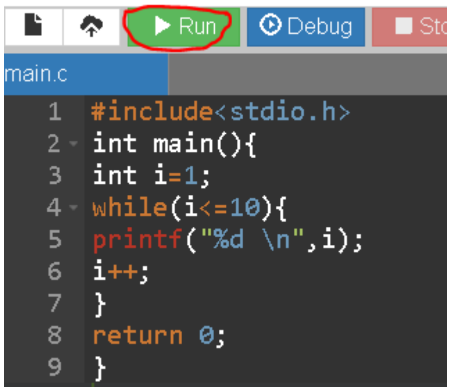
The syntax of while loop in c language is given below:

```
while(condition){

//code to be executed

}
```

Flowchart of while loop in C:

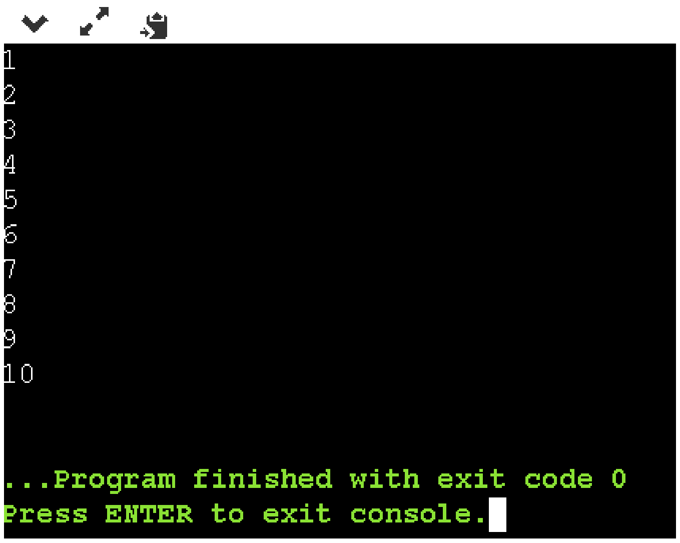


LAB: Simple program of while loop that prints table of 1.

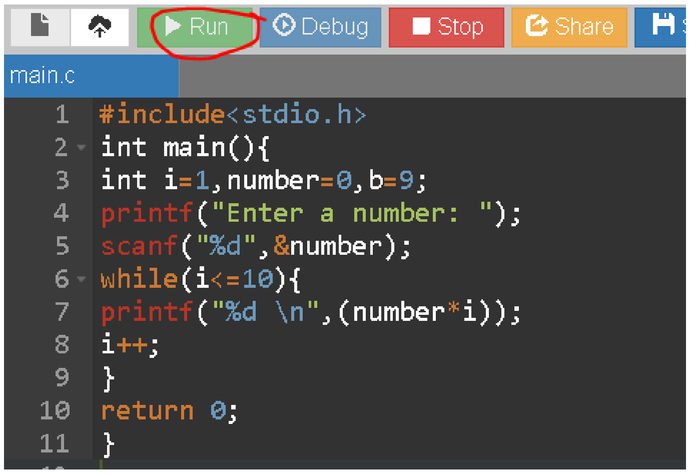


```
#include<stdio.h>
int main(){
int i=1;
while(i<=10){
printf("%d \n",i);
i++;
}
return 0;
}
```

Result/Output:

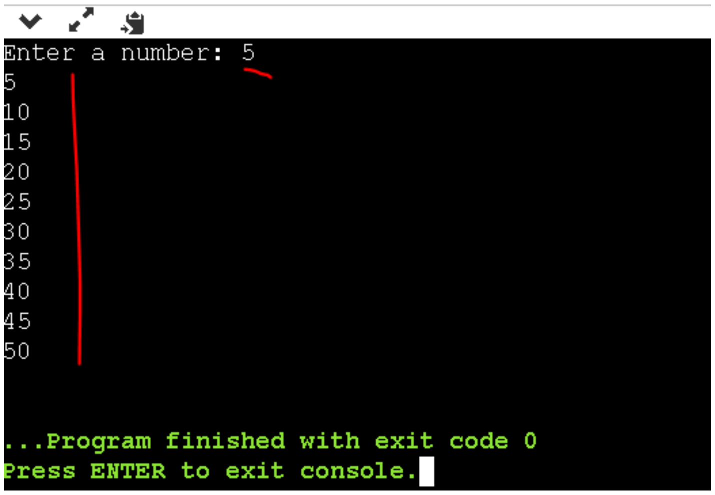

```
1
2
3
4
5
6
7
8
9
10

...Program finished with exit code 0
Press ENTER to exit console.
```

LAB: Program to print table for the given number using while loop in C:



```c
#include<stdio.h>
int main(){
int i=1,number=0,b=9;
printf("Enter a number: ");
scanf("%d",&number);
while(i<=10){
printf("%d \n",(number*i));
i++;
}
return 0;
}
```

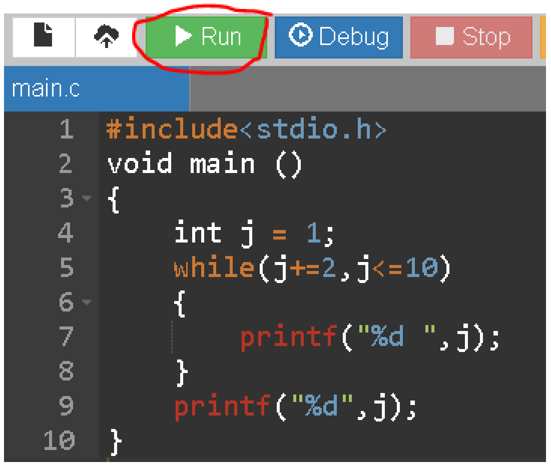
Result/Output:


```
Enter a number: 5
5
10
15
20
25
30
35
40
45
50

...Program finished with exit code 0
Press ENTER to exit console.
```

Properties of while loop

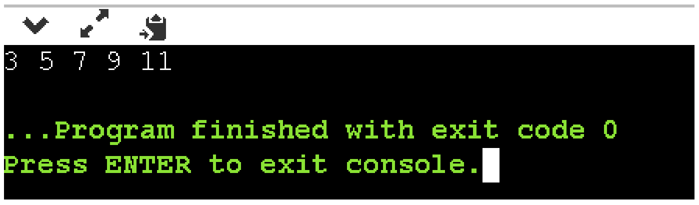
- A conditional expression is used to check the condition. The statements defined inside the while loop will repeatedly execute until the given condition fails.
- The condition will be true if it returns 0. The condition will be false if it returns any non-zero number.
- In while loop, the condition expression is compulsory.
- Running a while loop without a body is possible.
- We can have more than one conditional expression in while loop.
- If the loop body contains only one statement, then the braces are optional.

LAB: See the following example:



```c
#include<stdio.h>
void main ()
{
     int j = 1;
     while(j+=2,j<=10)
     {
          printf("%d ",j);
     }
     printf("%d",j);
}
```

Result/Output:



```
3 5 7 9 11

...Program finished with exit code 0
Press ENTER to exit console.
```

Infinitive while loop in C

If the expression passed in while loop results in any non-zero value then the loop will run the infinite number of times.

```c
while(1){

//statement

}
```

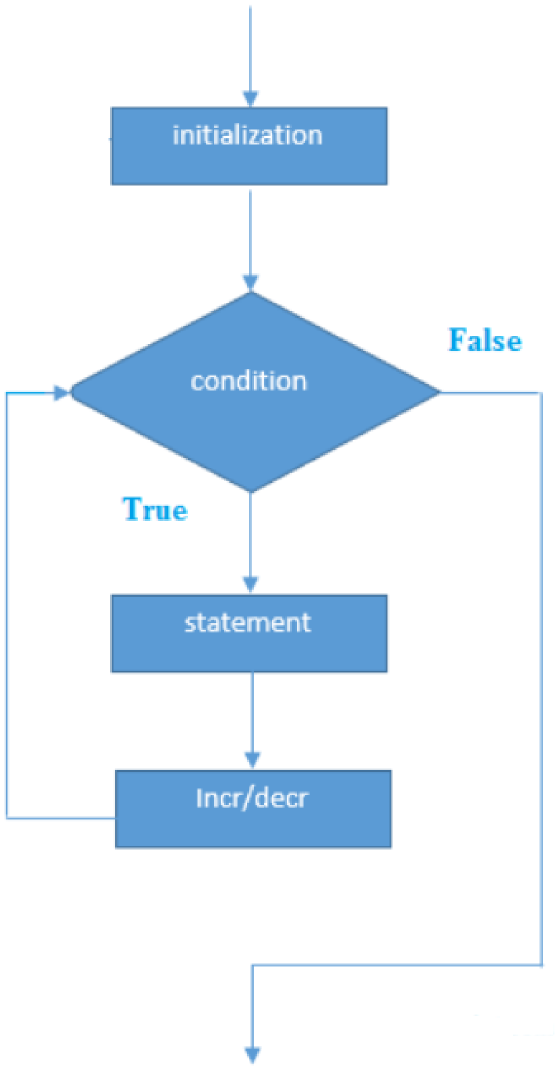
IV. C for loop

The for loop in C language is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

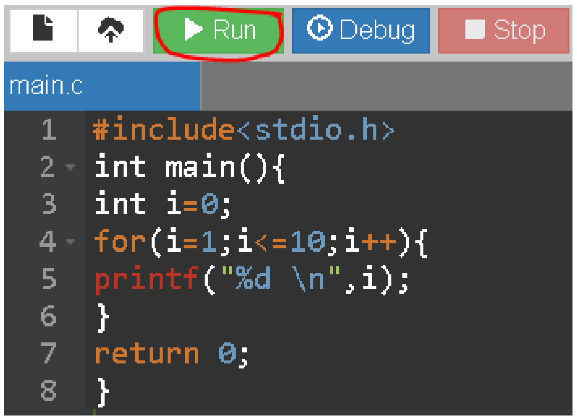
The syntax of for loop in c language is given below:

```
for(Expression 1; Expression 2; Expression 3){

//code to be executed

}
```

Flowchart of for loop in C:

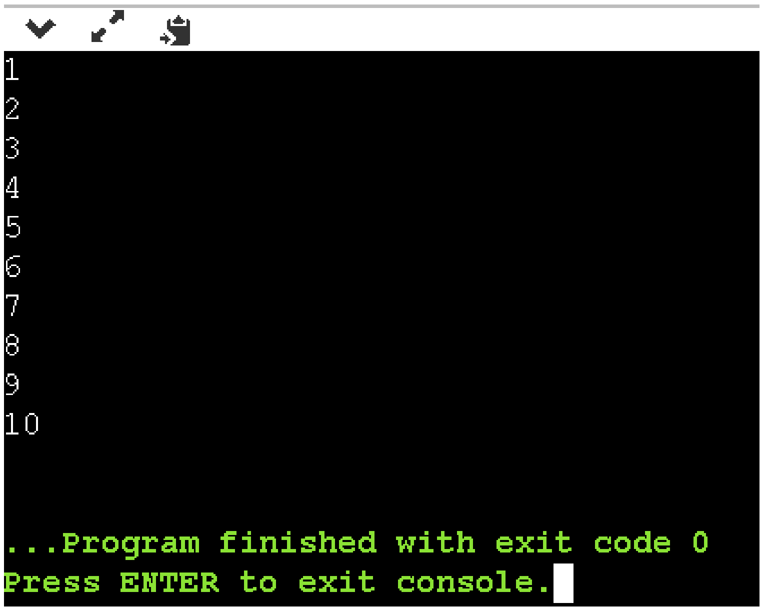


LAB: Simple program of for loop that prints table of 1.



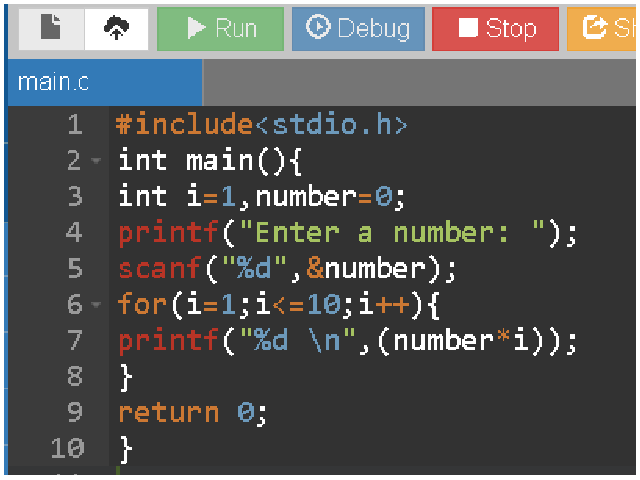
```c
#include<stdio.h>
int main(){
int i=0;
for(i=1;i<=10;i++){
printf("%d \n",i);
}
return 0;
}
```

Result/Output:



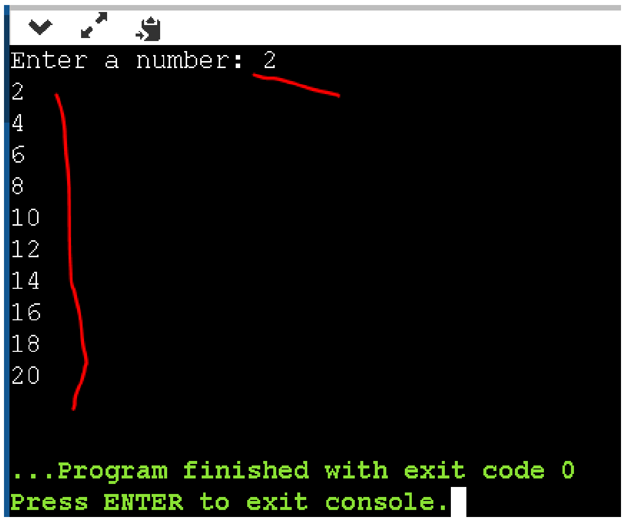
```
1
2
3
4
5
6
7
8
9
10


...Program finished with exit code 0
Press ENTER to exit console.
```

LAB: Print table for the given number using C for loop



```c
#include<stdio.h>
int main(){
int i=1,number=0;
printf("Enter a number: ");
scanf("%d",&number);
for(i=1;i<=10;i++){
printf("%d \n",(number*i));
}
return 0;
}
```

Result/Output:

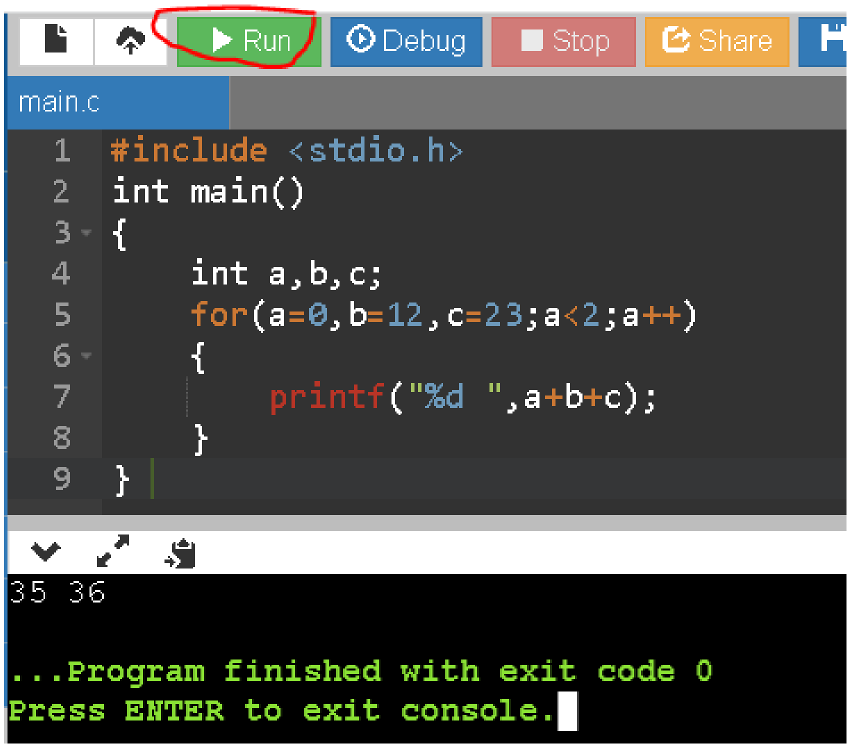


```
Enter a number: 2
2
4
6
8
10
12
14
16
18
20


...Program finished with exit code 0
Press ENTER to exit console.
```

Properties of Expression 1

- The expression represents the initialization of the loop variable.
- We can initialize more than one variable in Expression 1.
- Expression 1 is optional.
- In C, we can not declare the variables in Expression 1. However, It can be an exception in some compilers.

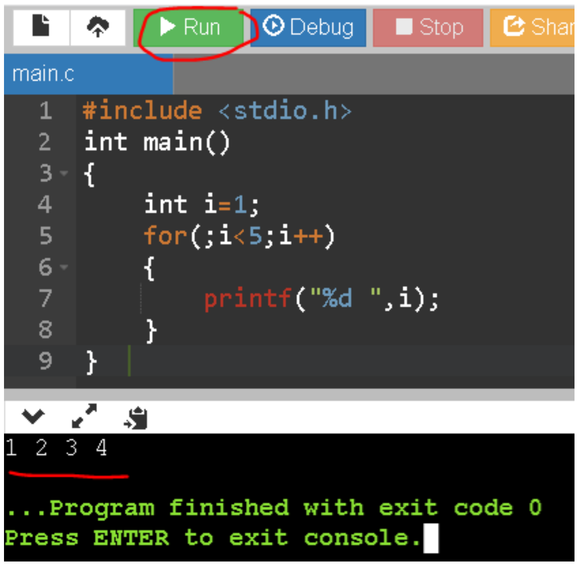
LAB: Example 1:



```c
#include <stdio.h>
int main()
{
    int a,b,c;
    for(a=0,b=12,c=23;a<2;a++)
    {
        printf("%d ",a+b+c);
    }
}
```

```
35 36

...Program finished with exit code 0
Press ENTER to exit console.
```

LAB: Example 2:



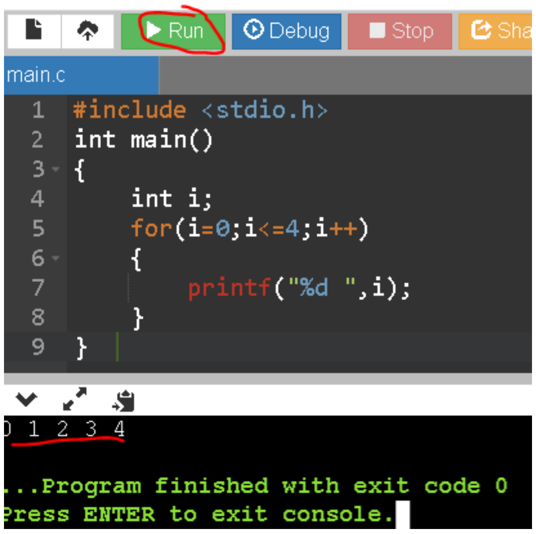
```c
#include <stdio.h>
int main()
{
    int i=1;
    for(;i<5;i++)
    {
        printf("%d ",i);
    }
}
```

```
1 2 3 4

...Program finished with exit code 0
Press ENTER to exit console.
```

Properties of Expression 2

- Expression 2 is a conditional expression. It checks for a specific condition to be satisfied. If it is not, the loop is terminated.
- Expression 2 can have more than one condition. However, the loop will iterate until the last condition becomes false. Other conditions will be treated as statements.
- Expression 2 is optional.
- Expression 2 can perform the task of expression 1 and expression 3. That is, we can initialize the variable as well as update the loop variable in expression 2 itself.
- We can pass zero or non-zero value in expression 2. However, in C, any non-zero value is true, and zero is false by default.

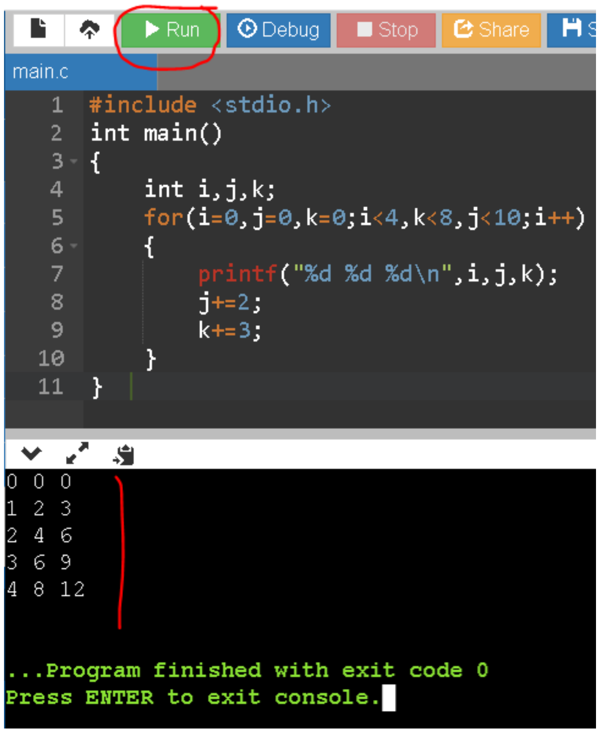
LAB: Example 1:



```c
#include <stdio.h>
int main()
{
    int i;
    for(i=0;i<=4;i++)
    {
        printf("%d ",i);
    }
}
```

```
0 1 2 3 4

...Program finished with exit code 0
Press ENTER to exit console.
```

LAB: Example 2:



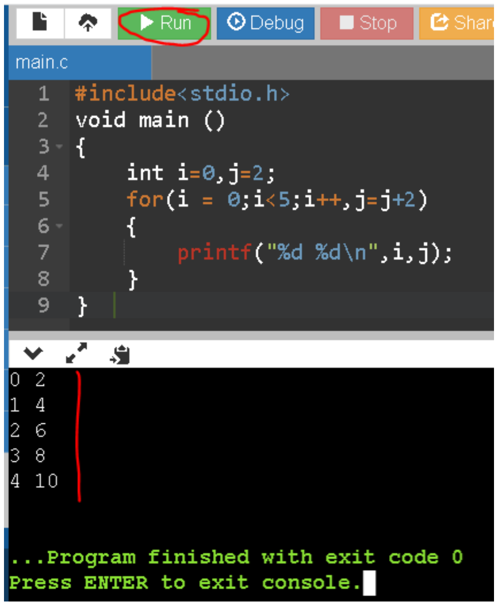
```c
#include <stdio.h>
int main()
{
    int i,j,k;
    for(i=0,j=0,k=0;i<4,k<8,j<10;i++)
    {
        printf("%d %d %d\n",i,j,k);
        j+=2;
        k+=3;
    }
}
```

```
0 0 0
1 2 3
2 4 6
3 6 9
4 8 12


...Program finished with exit code 0
Press ENTER to exit console.
```

Properties of Expression 3

- Expression 3 is used to update the loop variable.
- We can update more than one variable at the same time.
- Expression 3 is optional.

LAB: Example 1:

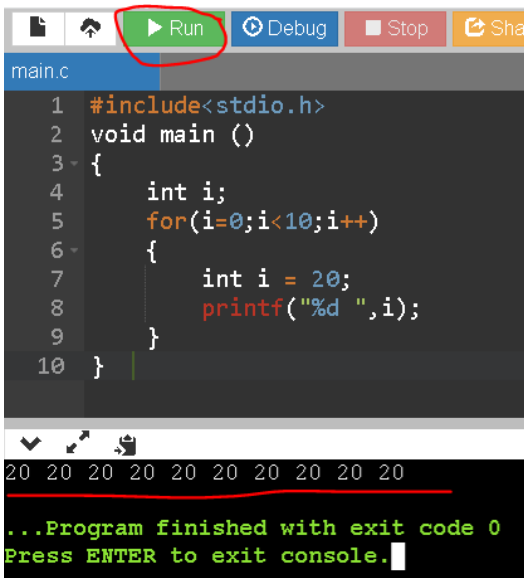


```c
#include<stdio.h>
void main ()
{
    int i=0,j=2;
    for(i = 0;i<5;i++,j=j+2)
    {
        printf("%d %d\n",i,j);
    }
}
```

```
0 2
1 4
2 6
3 8
4 10


...Program finished with exit code 0
Press ENTER to exit console.
```

Loop body

The braces {} are used to define the scope of the loop. However, if the loop contains only one statement, then we don't need to use braces. A loop without a body is possible. The braces work as a block separator, i.e., the value variable declared inside for loop is valid only for that block and not outside. Consider the following example.

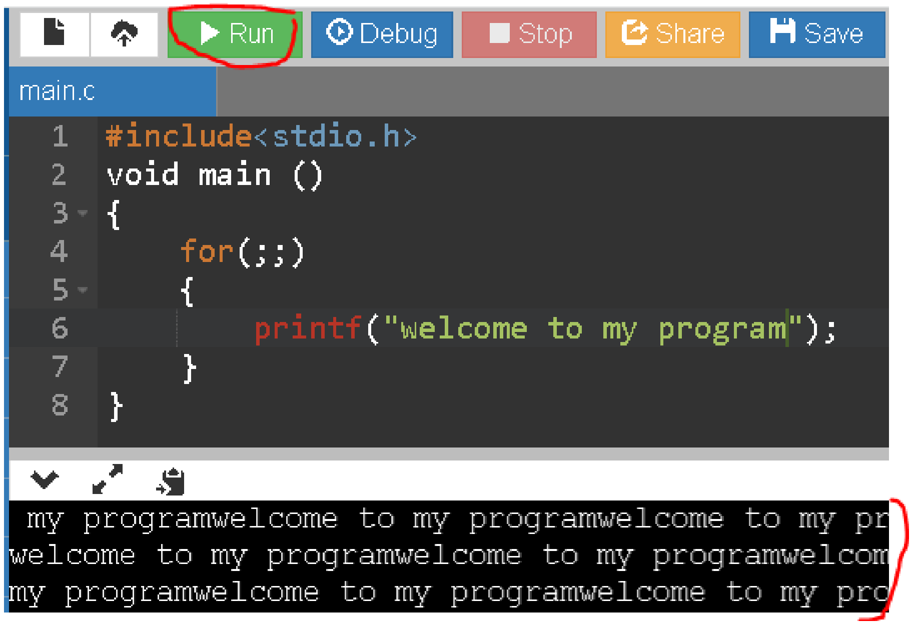


```c
#include<stdio.h>
void main ()
{
    int i;
    for(i=0;i<10;i++)
    {
        int i = 20;
        printf("%d ",i);
    }
}
```

```
20 20 20 20 20 20 20 20 20 20

...Program finished with exit code 0
Press ENTER to exit console.
```

Infinitive for loop in C

To make a for loop infinite, we need not give any expression in the syntax. Instead of that, we need to provide two semicolons to validate the syntax of the for loop. This will work as an infinite for loop.



```c
#include<stdio.h>
void main ()
{
    for(;;)
    {
        printf("welcome to my program");
    }
}
```

```
 my programwelcome to my programwelcome to my pr
welcome to my programwelcome to my programwelcom
my programwelcome to my programwelcome to my pro
```

If you run this program, you will see above statement infinite times.