```c
#include <stdio.h>

int main()
{
   int i=12,j;

   //printf returns no of characters "printed" by the printf function
   j=printf("%d%d",i, i);
   printf("%3d",j);

   j=printf("%d %d",i, i);
   printf("%3d",j);

   j=printf("%2d %3d",i, i);
   printf("%3d",j);

   int i=123,j;
   j=printf("%2d",i);
   printf("%d",j);

   //Scanf return no of inputs "received" by the scanf function
   j=scanf("%d%d",&i,&i);
   printf("%3d",j);
  return 0;
}
```

```
// Loop   "repetition"
// when we try to "do something more than one times"
#include <stdio.h>
int main()
{
   //Execution Style of FOR loop
       1    2     3
   for(i=0; i<10; i++)     // increment
     printf("print me");

   // 1 2
   // 3 2
   // 3 2
   // 3 2
   // 3  // exit from the loop

   // Output:   0 1 2 3 4 5 6 7 8 9

   //---------------------------
   for(i=0; i<10; i++)      //one statement => No loop braces
     printf("print me");

   //---------------------------
   for(i=0; i<10; i++)     //more than one statement => use loop braces
   {
     printf("print me");
     printf("print me");
   }

   //---------------------------
   for(i=10; i>0; i--)     // decrement
     printf("print me");

  //---------------------------
   for(i=0; i<10; i++)     // increment
     printf("print me");
```

```
   // for(initialization; checking/decision;  increment/decrement)
  //---------------------------
   i=0;                // initialization "outside"
   for(; i<10; i++)
     printf("print me");

   //---------------------------
   for(i=0; ;i++)
   {
      if(i<10)         // decisions "inside Loop"
      printf("print me");
   }

   //---------------------------
   for(i=0; i<10;)
   {
      printf("print me");
      i++;             // increment/decrement "inside Loop"
   }

   //---------------------------
   i=0;                // initialization "outside"
   for(; ;)            // this for loop is valid too
   {
      if(i<10)         // decisions "inside Loop"
        printf("print me");

      i++;             // increment/decrement "inside Loop"
   }



   return 0;
}
```

```
// Loop   "repetition"
// when we try to "do something more than one times"
#include <stdio.h>
int main()
{
   //Execution Style of FOR loop
       1    2     3
   for(i=0; i<10; i++)     // increment
     printf("print me");

   // 1 2
   // 3 2
   // 3 2
   // 3 2
   // 3  // exit from the loop

   // Output:   0 1 2 3 4 5 6 7 8 9

   //---------------------------
   for(i=0; i<10; i++)      //one statement => No loop braces
     printf("print me");

   //---------------------------
   for(i=0; i<10; i++)     //more than one statement => use loop braces
   {
     printf("print me");
     printf("print me");
   }

   //---------------------------
   for(i=10; i>0; i--)     // decrement
     printf("print me");

  //---------------------------
   for(i=0; i<10; i++)     // increment
     printf("print me");
```

```
   // for(initialization; checking/decision;  increment/decrement)
  //---------------------------
   i=0;                // initialization "outside"
   for(; i<10; i++)
     printf("print me");

   //---------------------------
   for(i=0; ;i++)
   {
      if(i<10)         // decisions "inside Loop"
      printf("print me");
   }

   //---------------------------
   for(i=0; i<10;)
   {
      printf("print me");
      i++;             // increment/decrement "inside Loop"
   }

   //---------------------------
   i=0;                // initialization "outside"
   for(; ;)            // this for loop is valid too
   {
      if(i<10)         // decisions "inside Loop"
        printf("print me");

      i++;             // increment/decrement "inside Loop"
   }



   return 0;
}
```

```
// WHILE LOOP

#include <stdio.h>
int main()
{

   //--------------------------
   i=0;              // Basic WHILE Loop: Increment
   while(i<10)
   {
      printf("print me");
      i++;
   }

   //--------------------------
   i=0;              //  WHILE Loop having ; => Outputs Nothing
   while(i<10);
   {
      printf("print me");
      i++;
   }

   //--------------------------
   i=10;             //Basic WHILE Loop: Decrement
   while(i>0)
   {
      printf("print me");
      i--;
   }

  //--------------------------
   i=8;             //Basic WHILE Loop with Break Statement
   while()          //def. of Break: exits from the current loop or go to the } (closing brace) of nearest loop
   {
      if(i>10)
```

```
        break;
      printf("print me");
      i++;
   }


  //---------------------------
   i=0;            // "nested" WHILE Loop without Break Statement
   while(i<10)
   {
      j=0;
      while(j<5)
      {
         k=0;
         while(k<3)
         {
            printf("print K");
            k++;
         }
         j++;
      }
      i++;
   }

  // How many times?    10 times  * 5 times * 3 times = 150 times "PRINT K"

   //---------------------------
   i=0;            // WHILE Loop with Break Statement
   while(i<10)
   {
      j=0;
      while(j<5)
      {
         k=0;
         while(k<3)
```

```
        {
            if(k==0) //def. of Break: exits from the current loop or just after the } (closing brace) of nearest loop
                break;
            printf("print K");
            k++;
        }
        j++;
    }
    i++;
}

//---------------------------
int i=0;                // Find the Output...
while(i<=32767)
{
    printf("%d",i);
    i++;
}

// Output:
//    0 1 2 3 4 5  .... 32766  32767   ++
//   -32768 -32767 -32766 .... 0 .... 32767
//  -32768 .... 0 ... 32767
//  and so on ....

// Explanation ??   WHY an interger supports any number within a limit of -32768  to +32767
//    an integer variable (for a 32 bit compiler) takes 2B (B bytes b bits)
// 2B = 16b (b bits => 'bi'nary digi't')   => 2^16 combinations are possible
//  65536 combinations =>   32768 nos are used for +ve and 32768 nos are used for -ve
//  so the range will be -32768 ... 0 ... +32767 (range of an int variable)
```

//Character Variable : 1B (32 bit compiler) 2B (64bit compiler)
// 8b => 2^8 combinations => 256 combinations => -128 to +127 is the range of a cahracter variable

// Float ? 4B => 32 bit => 2^32 combinations => HOME TASK

//----------
// PRE INCREMENT / POST INCREMENT OPERATOR

```
int a=2;
a=a+1;  or     a++;  or     a+=1;

int a=2;
a=a-1;  or     a--;  or     a-=1;

int a=2;
a=a*1;  or     a*=1;    //  cant write this one like a**;

int a=2;
a=a/1;  or     a/=1;    //  cant write this one like a//;
```

//---------- Basic Use of

```
                          a  b  c
int a=2, b, c;                2  ?  ?
b= a++;   //post increment      3  2  ?
c= ++a;   //pre increment       4  2  4

                          a  b  c
int a=2, b, c;                2  ?  ?
b= a++;   //post increment      3  2  ?
c= ++b;   //pre increment       3  3  3

                          a  b  c
```

```
    int a=2, b, c;                  2  ?  ?
    b= ++a;   //pre increment         3  3  ?
    c= a++;   //post increment        4  3  3

                                a  b  c
    int a=2, b, c;                  2  ?  ?
    b= ++a * ++a;                     4  16 ?
    b= a++ * a++;                     6  16 ?

                                a  b  c
    int a=2, b, c;                  2  ?  ?
    b= ++a * a++;                     4  9  ?

    //---------------------------
    i=0;              // Pre Increment operator
    while(++i<=10)
      printf("%d",i);     Output?  1 2 3 ...

    //---------------------------
    i=0;              // Post Increment operator
    while(i++<10)
      printf("%d",i);    Output?  1 2 3 ...

    //---------------------------
    i=10;              // Post Increment operator: Shortest Condition
    while(i--)
      printf("%d",i);    //Find the Output ?   9 ... 0 will be the output

    return 0;
}
```
