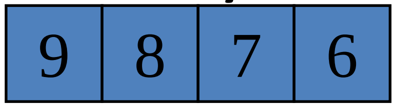
Some tricky points

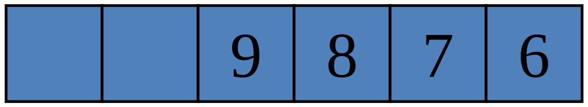
Assignment: Practice on the your machine

- printf(“%d”,9876)



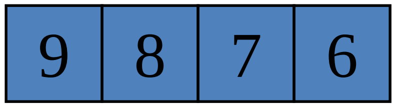
```
9876
```

- printf(“%6d”,9876)



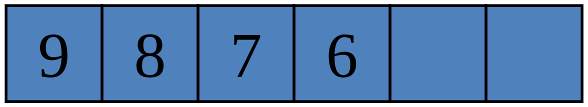
```
  9876
```

- printf(“%2d”,9876)



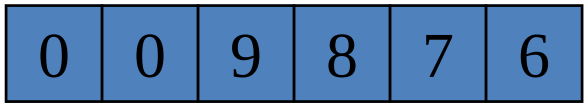
```
9876
```

- printf(“%-6d”,9876)



```
9876  
```

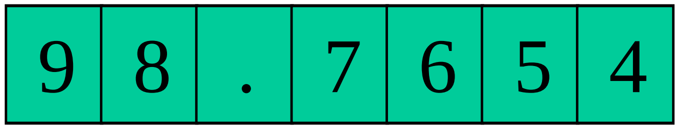
- printf(“%06d”,9876)



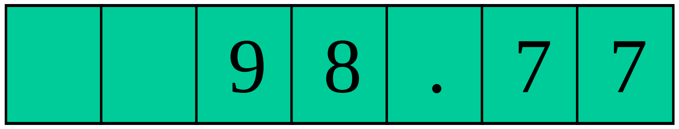
```
009876
```

Example

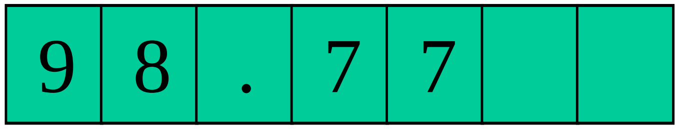
printf(“%7.4f”,98.7654)



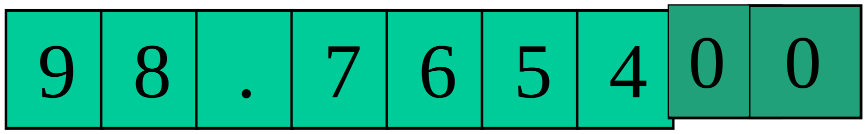
printf(“%7.2f”,98.7654)



printf(“%-7.2f”,98.7654)



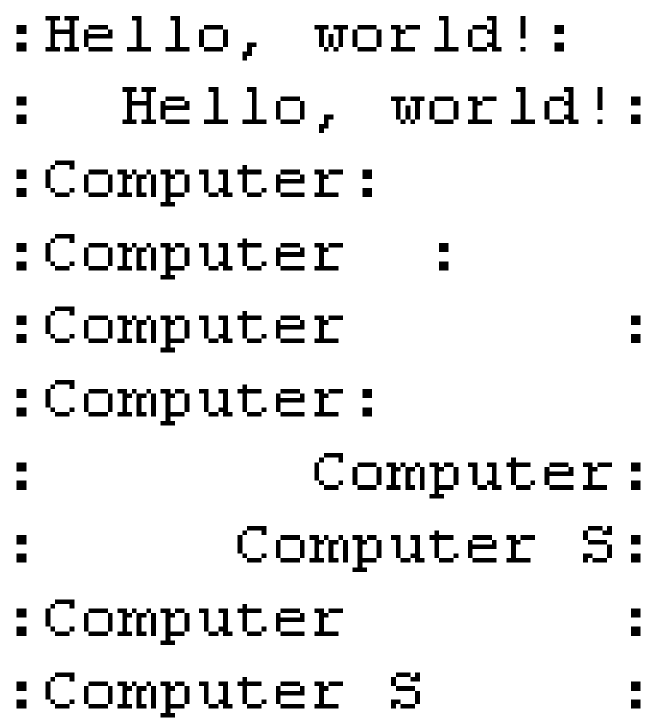
printf(“%f”,98.7654)



Note:-Using precision in a conversion specification in the format control string of a scanf statement is wrong.

Example

```
#include<stdio.h>
main()
{
   printf(":%s:\n", "Hello, world!");
     printf(":%15s:\n", "Hello, world!");
     printf(":%.10s:\n", "Computer");
     printf(":%-10s:\n", "Computer");
     printf(":%-15s:\n", "Computer");
     printf(":%.15s:\n", "Computer");
     printf(":%15.10s:\n", "Computer");
   printf(":%15.10s:\n", "Computer Science");
     printf(":%-15.10s:\n", "Computer");
   printf(":%-15.10s:\n", "Computer Science");
}
```



```
:Hello, world!:
:  Hello, world!:
:Computer:
:Computer  :
:Computer       :
:Computer:
:       Computer:
:     Computer S:
:Computer       :
:Computer S     :
```

Output?

```
#include <stdio.h>
int main()
{
   float c = 5.0;
   printf ("Temp in Fahrenheit is %.2f", (9/5)*c + 32);
   return 0;
}
```

Since 9 and 5 are integers, integer arithmetic happens in subexpression (9/5) and we get 1 as its value.

37.00 answer

Output?

```
#include <stdio.h>
int main()
{
   char a = '\012';

   printf("%d", a);

   return 0;
}
```

Explanation: The value ‘\012’ means the character with value 12 in octal, which is decimal 10

Output?

```
#include<stdio.h>
int main()
{
    float x = 0.1;
    if ( x == 0.1 )
        printf("IF");
    else if (x == 0.1f)
        printf("ELSE IF");
    else
        printf("ELSE");
}
```

Ans : ELSE IF

Output?

```
#include<stdio.h>

int main()
{
  int c;
  printf(“study for %nexams ", &c);
  printf("%d", c);
  getchar();
  return 0;
}
```

%n

In C printf(), %n is a special format specifier which instead of printing something causes printf() to load the variable pointed by the corresponding argument with a value equal to the number of characters that have been printed by printf() before the occurrence of %n.

Answer is : “study for exams 10”

Output?

```
#include <stdio.h>

int main()
{
  printf(" \“study %% FOR %% exams\"");
  getchar();
  return 0;
}
```

(A) “study % FOR % exams”
(B) study % FOR % exams
(C) \” study %% FOR %% exams \”
(D) study %% FOR %% exams

Answer: (A)

Output?

```
#include <stdio.h>
// Assume base address of " StudyExam " to be 1000
int main()
{
   printf(5 + “StudyExam");
   return 0;
}
```

(A) StudyExam
(B) Exam
(C) 1005
(D) Compile-time error

Answer: (B)

Explanation:
printf is a library function defined under *stdio.h* header file. The compiler adds 5 to the base address of the string through the expression ***5 + “StudyExam”*** .

6Then the string “Exam” gets passed to the standard library function as an argument.

Output?

```
#include <stdio.h>
int main(void)
{
  int x = printf(“StudyExam");
  printf("%d", x);
  return 0;
}
```

Run on IDE

(A) StudyExam 9
(B) StudyExam 10
(C) StudyExam StudyExam
(D) StudyExam 1

Answer: (A)

Explanation: The printf function returns the number of characters successfully printed on the screen. The string “StudyExam” has 9 characters, so the first printf prints StudyExam and returns 9.

Output?

```
#include<stdio.h>
int main()
{
    printf("%d", printf("%d", 1234));
    return 0;
}
```

(A) 12344
(B) 12341
(C) 11234
(D) 41234

Answer: (A)

Short-Circuiting in Logical Operators:

In case of **logical AND**, the second operand is not evaluated if first operand is false..

```
#include <stdio.h>
#include <stdbool.h>

int main()
{
   int a=10, b=4;
   bool res = ((a == b) && printf(“Hello"));
   return 0;
}
```

For example, given program doesn’t print “Hello” as the first operand of logical AND itself is false

output?

```
#include <stdio.h>
#include <stdbool.h>
int main()
{
    int a=10, b=4;
    bool res = ((a != b) && printf(“Hello"));
    return 0;
}
```

In case of **logical OR**, the second operand is not evaluated if first operand is true

```
#include <stdio.h>
#include <stdbool.h>
int main()
{
    int a=10, b=4;
    bool res = ((a != b) ||
    printf(“hello"));
    return 0;
}
```

```
#include <stdio.h>
#include <stdbool.h>
int main()
{
    int a=10, b=4;
    bool res = ((a == b) ||
    printf(" hello"));
    return 0;
}
```

OUTPUT?

```
#include <stdio.h>

int main()
{
  int a = 1;
  int b = 1;
  int c = a || --b;
  int d = a-- && --b;
  printf("a = %d, b = %d, c = %d, d = %d", a, b, c, d);
  return 0;
}
```

a = 0, b = 0, c = 1, d = 0

Sizeof operator

sizeof() operator is used in different way according to the operand type.

1. When operand is a Data Type.

When *sizeof()* is used with the data types such as int, float, char… etc it simply return amount of memory is allocated to that data types.

Let see example:

```
#include<stdio.h>
int main()
{
   printf("%d\n",sizeof(char));
   printf("%d\n",sizeof(int));
   printf("%d\n",sizeof(float));
   printf("%d", sizeof(double));
   return 0;
}
```

```
#include <stdio.h>
int main()
{
   //Assume sizeof character is 1 byte and sizeof
   integer is 4 bytes
   printf("%d", sizeof(printf("GeeksQuiz")));
   return 0;
}
```

2. When operand is an expression

When *sizeof()* is used with the expression, it return size of the expression. Let see example:

```
#include<stdio.h>
int main()
{
   int i = 0;
   double d = 10.21;
   printf("%d", sizeof(i+d));
   return 0;
}
```

Comma operator

```
// PROGRAM 1
#include<stdio.h>

int main(void)
{
   int a = 1, 2, 3;
   printf("%d", a);
   return 0;
}
```

Compile time error

```
// PROGRAM 2
#include<stdio.h>

int main(void)
{
   int a;
   a = 1, 2, 3;
   printf("%d", a);
   return 0;
}
```

Answer : 1

```
// PROGRAM 3
#include<stdio.h>

int main(void)
{
   int a;
   a = (1, 2, 3);
   printf("%d", a);
   return 0;
}
```

- Comma works just as a separator in PROGRAM 1 and we get compilation error in this program.
- Comma works as an operator in PROGRAM 2.
- Precedence of comma operator is least in operator precedence table. So the assignment operator takes precedence over comma and the expression “a = 1, 2, 3″ becomes equivalent to “(a = 1), 2, 3″.
- That is why we get output as 1 in the second program.

- In PROGRAM 3, brackets are used so comma operator is executed first and we get the output as 3

Output?

```
#include<stdio.h>

int main(void)
{
     int a=0;
     int b=20;
     char x=1;
     char y=10;
     if (a,b,x,y)
     printf(“hello”);

    return 0;
}
```

Answer : hello

Output?

```
int main()
 {
int a = 3, b = -8, c = 2;
printf("%d", a % b / c);
 return 0;
}
```

The output is 1.

% and / have same precedence and left to right associativity. So % is performed first which results in 3 and / is performed next resulting in 1.
The emphasis is,***sign of left operand is appended to result in case of modulus operator in C***.

Output?

NOTE: This output is compiler and machine dependent

```
int main()
 {
 int a=0;
printf("%d %d %d", ++a, ++a, ++a);
 return 0;
}
```

```
int main()
 {
 int a=2;
    int b=3;
printf("%d ", ++a + b++);
 return 0;
}
```

3 3 3

Ans 6

Output?

NOTE: This output is compiler and machine dependent

```
int main()
 {
    int i,j,k;
    i=0;
printf("%d %d %d", ++k, k=++i + ++j, j=++i);
 return 0;
}
```

```
5   5   2
```

Output?

NOTE: This output is compiler and machine dependent

```
int main()
 {
int i=5;
printf(“%d\n%d\n%d\n%d\n%d”,i++,i--,++i,--i,i);
}
```

```
4
5
5
5
5
```

Output?

```
int main()
 {
int a=1,b=2,c=3;
C+=(a>0&&a<=10)?++a : a/b;

printf(“%d %d %d”,a,b,c);
}
```

```
2   2   5
```

```
#include <stdio.h>
int main(void)
 {
int a = 1;
 int b = 1;
  int c;
    printf("a = %d, b = %d, c = %d", a, b, a||--b);
    return 0;
}
```

```
1 1 1
```