

MINOR ASSIGNMENT-00

Practical Programming with C (CSE 3544)

Publish on: 23-09-2025

Course Outcome: CO₁

Submission on: 27-09-2025

Learning Level: L₂

Program Outcome: PO₁

Problem Statement:

Experiment with general form of C program, format string, format specifier, formatted output function-*printf()* and formatted input function-*scanf()*.

1. Write a C program to display the following message on the monitor.

```
This is my First C Program!
I konw:
    where to write C program,
    how to save and edit the program!
To compile- (1) gcc filename.c
            (2) gcc filename.c -o myout
To run-    (1) ./a.out
            (2) ./myout
-----
Able to run successfully !!!
```

Write your program here

```
#include <stdio.h>

int main() {
    printf("This is my First C Program!\n");
    printf("I konw:\n");
    printf("where to write C program,\n");
    printf("how to save and edit the program!\n");
    printf("To compile- (1) gcc filename.c\n");
    printf("                (2) gcc filename.c -o myout\n");
    printf("To run- (1) ./a.out\n");
    printf("        (2) ./myout\n");
    printf("-----\n");
    printf("Able to run successfully !!!\n");
    return 0;
}
```

2. Write the output of the code snippet that makes the use of the **printf** function.

```
int main() {
    float i=2.0, j=3.0;
    printf("%f %f %f", i, j, i+j);
    return 0;
}
```

Write/paste output in exact form as expected

2.000000 3.000000 5.000000

3. Express the output of the code snippet;

```
int main() {
    printf("%d==%f==%lf\n", 5, 55.5, 55.5);
    printf("%i==%e==%E\n", 5, 555.5, 123.45);
    printf("%o==%g==%G\n", 9, 555.5, 123.45);
    return 0;
}
```

Write/paste output in exact form as expected

5==55.500000==55.500000
5==5.555000e+02==1.234500E+02
11==555.5==123.45

4. State the output of the code snippet;

```
int main() {
    printf("%d==%i==%o==%x\n", 32, 32, 32, 32);
    printf("%d==%i==%#o==%#x\n", 32, 32, 32, 32);
    printf("%d==%i==%#o==%#X\n", 32, 32, 32, 32);
    printf("%+d==%+i==%#o==%#X\n", 32, 32, 032, 0x45b);
    return 0;
}
```

Write/paste output in exact form as expected

32 == 32 == 40 == 20
32 == 32 == 040 == 0x20
32 == 32 == 040 == 0x20
+32 == +32 == 032 == 0x45B

5. The given code snippet generate the same floating-point output in three different form. Mention the two different form into the space provided below the code snippet.

```
int main() {
    double x=3000.0, y=0.0035;
    printf("%f %f %f\n", x, y, x*y, x/y);
    printf("%e %e %e\n", x, y, x*y, x/y);
    printf("%E %E %E\n", x, y, x*y, x/y);
    return 0;
}
```

Write/paste output in exact form as expected

3000.000000 0.003500 10.500000 857142.857143
 3.000000e+03 3.500000e-03 1.050000e+01 8.571429e+05
 3.000000E+03 3.500000E-03 1.050000E+01 8.571429E+05

6. Assuming the **side**, and **area** are type **float** variables containing the length of one side in cm and area of a square in square cm, write a statement using **printf** that will display this information in this form:

The area of a square whose side length is _____ cm
 is _____ square cm.

Write/paste output in exact form as expected

printf("The area of a square whose side length is %.2f cm is
 %.2f square cm.\n", side, area);

7. Show the exact form of the output line when **n** is 345. (consider **_** = 1 blank).

```
printf("Three values of n are %4d*%5d*%d\n", n, n, n);
```

Write/paste output in exact form as expected

Three values of n are 345* 345*345

8. State the data types you would use to represent the following items: number of students in your section, a letter grade on the AD1 exam, average number of days in a semester, the name of the topper

of your class, total number of courses in this semester. Also specify the format specifier/placeholder for the variables used in the above case.

Specify answer in two column: data type and the required format specifier

int	%d
char	%c
float	%f
char[]	%s
int	%d

9. The following C code snippet illustrate the use of minimum field width feature in **printf** function. Write the output of the code snippet assuming _ as 1 blank space.

```
int main()
{
    int i=54321;
    float x=876.543;
    printf("%3d: %5d: %10d: %12d:\n", i, i, i, i);
    printf("%3f: %10f: %13f: %f:\n", x, x, x, x);
    return 0;
}
```

State the output in exact form

```
:54321: :54321: : 54321: : 54321:
:876.543000: : 876.543000: : 876.543000: :876.543000:
```

10. The following C code snippet illustrate the use of minimum field width feature in **printf** function. Write the output of the code snippet assuming _ as 1 blank space.

```
int main()
{
    int i=54321;
    float x=876.543;
    printf("%-3d: %-5d: %-10d: %-12d:\n", i, i, i, i);
    printf("%-3f: %-10f: %-13f: %f:\n", x, x, x, x);
    return 0;
}
```


State the output in exact form

```
%5422\ : %54321 : %54321 : : %54321 :
%276.542000 : %276.542000 : %276.542000 : : %276.543000 :
```

11. The following C code snippet illustrate the use * as minimum field width feature in **printf** function. Write the output of the code snippet assuming _ as 1 blank space.

```
int main()
{
    int ivar=1234;
    printf(":%d:\n",10,ivar);
    printf(":%-d:\n",10,ivar);
    return 0;
}
```

State the output in exact form

```
 : 1234:
 :1234 :
```

12. The following C code snippet illustrate the use * as minimum field width feature in **printf** function. Write the output of the code snippet assuming _ as 1 blank space.

```
int main()
{
    int ivar=1234;
    printf(":%*.d:\n",10,4,ivar);
    printf(":%-*.d:\n",10,4,ivar);
    return 0;
}
```

State the output in exact form

```
 : 1234:
 :1234 :
```

13. The following C code snippet illustrate the use + as minimum field width feature in **printf** function. Write the output of the code snippet assuming _ as 1 blank space.

```
int main()
{
    int ivar=1234;
    printf(":%+.7d:\n", 13, 7, ivar);
    printf(":%-+.7d:\n", 13, 7, ivar);
    return 0;
}
```

State the output in exact form

```
: 0001234:
:0001234 :
```

14. The following C code snippet illustrate the use * as minimum field width feature in **printf** function. Write the output of the code snippet assuming _ as 1 blank space.

```
int main()
{
    int ivar=1234;
    printf(":%*.7d:\n", 7, ivar);
    printf(":%-*.7d:\n", 7, ivar);
    return 0;
}
```

State the output in exact form

```
:0001234:
:0001234:
```

15. The following code snippet shows a case without minimum field width specification, but with precision specification. Write the desired output.

```
int main()
{
    float x=123.456;
    printf("%f %.3f %.1f %.0f\n", x, x, x, x);
    printf("%e %.5e %.3e %.0e\n", x, x, x, x);
    return 0;
}
```

State the output in exact form

123.456000 123.456 123.5 123
1.234560e+02 1.23456e+02 1.235e+02 1e+02

16. The minimum field width and precision in the format string of printf function can be applied to character data as well as numerical data. When applied to a string, the minimum field width is interpreted in the same manner as with the numerical quantity. However, the precision specification will determine the **maximum** number of characters that can be displayed. If the precision specification is less than the total number of characters in the string, the excess right-most characters will not be displayed. This will occur even if the minimum field width is larger than the entire string, resulting in the addition of leading blanks to the truncated string. So, write the output of the following code snippet;

```
int main()
{
    char line[]="hexadecimal";
    printf(":%10s: :%15s: :%15.5s: :%.5s:\n",line,line,line,line);
    return 0;
}
```

State the output in exact form

:hexadecimal: : hexadecimal: : hexad: : hexad:

17. Determine the output of the code snippet that uses the uppercase conversion characters in the printf function.

```
int main()
{
    int a=0x80ec;
    float b=0.3e-12;
    printf(":%#4x: :%#10.2e:\n",a,b);
    printf(":%#4X :%#10.2E:\n",a,b);
    return 0;
}
```

State the output in exact form

: 0x80ec: : 3.00e-13:

: 0x80EC : 3.00E-13:

18. The following program shows the placement of **flags**(i.e -, +, 0, space, #) in printf format string just after the symbol % to get some specific affects in the appearance of the printf output.

```
int main(){
    int i=345;
    float x=34.0, y=-5.6;
    printf(":%6d: :%7.0f: :%10.1e:\n",i,x,y);
    printf(":%-6d: :%-7.0f: :%-10.1e:\n",i,x,y);
    printf(":%+6d: :%+7.0f: :%+10.1e:\n",i,x,y);
    printf(":%-+6d: :%-+7.0f: :%-+10.1e:\n",i,x,y);
    printf(":%6.0d: :%#7.0f: :10g: :%#10g:\n",x,x,y,y);
    return 0;}
```

State the output in exact form

: 345: : 34: : -5.6e+00:

:345 : :34 : : -5.6e+00:

: +345: : +34: : -5.6e+00:

:+345 : :+34 : : -5.6e+00 :

: 34: : 34. : :10g: : -5.6:

19. Predict the output of the given code snippet that uses the flags with unsigned decimal, octal and hexadecimal numbers.

```
int main(){
    int i=345, j=01767, k=0xa0bd;
    printf(":%8u: :%8o: :%8x:\n",i,j,k);
    printf(":%-8u: :%-8o: :%-8x:\n",i,j,k);
    printf(":%#8u: :%#8o: :%#8x:\n",i,j,k);
    printf(":%08u: :%0o0: :%08x:\n",i,j,k);
    printf(":% #8u: :% #8o: :% #8x:\n",i,j,k);}
```



```
return 0;
}
```

State the output in exact form

```
∴ 345: : 1777: : aobd:
∴ 345 : : 1777 : : aobd :
∴ 345: : 01777: : 0xaobd:
∴ 00000345: : 17770: : 0000aobd:
∴ 345: : 01777: : 0xaobd:
```

20. Predict the output of the given code snippet that outline the use of flags with string.

```
int main()
{
    char line[]="lower-case";
    printf(":%15s: :%15.5s: :%.5s:\n",line,line,line);
    printf(":%-15s: :%-15.5s: :%-5s:\n",line,line,line);
    return 0;
}
```

State the output in exact form

```
∴ lower-case: : lower: : lower:
∴ lower-case : : lower : : lower:
```

21. Predict the output of the given code snippet that illustrates how printed output can be labeled.

```
int main()
{
    float a=2.2, b=-6.2, x1=.005, x2=-12.88;
    printf("$%4.2f %7.1f%%\n",a,b);
    printf("x1=%7.3f x2=%7.3f\n",x1,x2);
    return 0;
}
```

State the output in exact form

\$2.20 -6.2%

X1 = 0.005 X2 = -12.880

22. Write a program to read three characters from the standard input device (i.e. keyboard) and display the characters on the standard output device (i.e. monitor) using %c format specifier/place holder. The different ways to provide input to the program are; (i) S O A (ii) S <enter> O <enter> A <enter> (iii) <multiple spaces> S <multiple spaces> O <multiple spaces> A <enter>. Redesign your program to use %s in scanf for the same objective instead of %c in scanf.

Write your code here

```
#include <stdio.h>
int main() {
    char ch1, ch2, ch3;
    printf("Enter three characters: ");
    scanf("%c %c %c", &ch1, &ch2, &ch3);
    printf("You entered: %c %c %c\n", ch1, ch2, ch3);
    return 0;
}

#include <stdio.h>
int main() {
    char str1[2], str2[2], str3[2];
    printf("Enter three characters: ");
    scanf("%1s %1s %1s", str1, str2, str3);
    printf("You entered: %s %s %s\n", str1, str2, str3);
    return 0;
}
```

23. Choose the output of the code snippet;

```
int main()
{
    int i=10, m=10;
    printf("%d", i*m, m);
    return 0;
}
```

State the output in exact form

(A) 100 10

(C) 10

~~(B) 100~~

(D) Error

Answer with reason:

100

Only the first argument i.e. m is used for the single %d, the extra m is ignored.

24. Predict the output of the given code snippet that illustrates a form of formatted input function scanf.

```
int main()
{
    int sr=100,pr=100;
    sr=scanf("Me a scanner");
    pr=printf("scanf returns=%d\n",sr);
    printf("printf returns::%d\n",pr);
    return 0;
}
```

State the output in exact form

(A) 100 100

~~(C) 0 16~~

(E) Compilation error

(B) 0 100

(D) 16 0

(F) Run-time error

Answer with reason:

scanf has no format specifiers, so it matches the literal string in input. It returns the number of input items successfully matched, which is 0.

printf prints the value which has 16 characters, so it returns 16.

25. Predict the output of the given code snippet;

```
int main()
{
    int num;
    printf("Enter a number:");
    scanf("%2d", &num);
    printf("number=%d", num);
    return 0;
}
```

State the output in exact form

Choose the output if inputs are (i) 2345 (ii) 9 (iii) 76 (iv) 456 on different run.

(A) 2345 9 76 456

(C) 456 76 9 2345

(B) 23 9 76 45

(D) No output

Answer with reason:

Input: 2345 \rightarrow %2d reads 23 \rightarrow num = 23 \rightarrow remaining 45 stays in buffer.

Input: 9 \rightarrow %2d reads 9 \rightarrow num = 9

Input: 76 \rightarrow %2d reads 76 \rightarrow num = 76

Input: 456 \rightarrow %2d reads 45 \rightarrow num = 45 \rightarrow remaining 6 stays in buffer.

26. Predict the output of the given code snippet;

```
int main()
{
    int num1=0, num2=0, num3=0;
    printf("Enter a number:");
    scanf("%2d%3d%4d", &num1, &num2, &num3);
    printf("%d %d %d", num1, num2, num3);
    return 0;
}
```

State the output in exact form

Choose the output, if inputs are (i) 2345 (ii) 9 (iii) 76 (iv) 456 on different runs.

(A) 2345 9 76 456

(C) 456 76 9 2345

(B) 23 9 76 45

(D) No output

Answer with reason:

Each %Nd in scanf reads at most N digits; remaining digits stay in the input buffer or are used by the next specifier, so numbers are split according to the specified widths.

27. Choose the output of the code snippet;

```
int main()
{
    int num1=0, num2=0, num3=0;
    printf("Enter the number as <345678>:");
    scanf("%1d%2d%3d", &num1, &num2, &num3);
    num1=num1+num2+num3;
    printf("%d\n", num1);
    return 0;
}
```


State the output in exact form

(A) 87654

(C) 726

(B) 345678

(D) No output

Answer with reason: $\%1d \rightarrow 3 \rightarrow \text{num1} = 3$ $\%2d \rightarrow 45 \rightarrow \text{num2} = 45$ $\%3d \rightarrow 678 \rightarrow \text{num3} = 678$ $\text{num1} = \text{num1} + \text{num2} + \text{num3} = 3 + 45 + 678 = 726$

28. Choose the output of the code snippet;

```
int main()
{
    int i=10,m=10;
    printf("%d",printf("%d %d ",i,m));
    return 0;
}
```

State the output in exact form

(A) 10 10 6

(C) 6 6 6

(B) 10 10 10

(D) No output

Answer with reason:

The inner printf prints 10 10 (6 characters) and returns 6, the outer printf then prints this return value.

29. A C program contains the following form; Suppose that the following string has been assigned to **text**

Programming with C can be a challenging creative activity. Show the output resulting from the following printf statements

```
int main()
{
    char text[100];
    :::::::::::
    printf("%s\n",text);
    printf("%18s\n",text);
    printf("%.18s\n",text);
    printf("%18.7s\n",text);
    printf("%-18.7s\n",text);
    return 0;
}
```

State the output in exact form

Programming with C can be a challenging creative activity.

Programming with C can be a challenging creative activity.

Programming with C

Program

Program

30. A C program contains the following statements. Write an appropriate **scanf** function to enter numerical values of **i**, **j** and **k** assuming

- (i) The values for **i**, **j** and **k** will be decimal numbers. Display the values.
- (ii) The value of **i** will be decimal integer, **j** an octal integer and **k** a hexadecimal integer. Display the values.
- (iii) The value of **i** and **j** will be hexadecimal number and **k** an octal integer. Display the values.

State the output in exact form

Code for (i)

```
scanf("%d %d %d", &i, &j, &k);
```

Output:

i=10, j=20, k=30

State the output in exact form

Code for (ii)

```
scanf("%d %o %x", &i, &j, &k);
```

Output:

i=10, j=15, k=10

State the output in exact form

Code for (iii)

```
scanf("%x %x %o", &i, &j, &k);
```

Output:

i=10, j=31, k=15

31. Describe the output of the code snippet;

```
int main(){
    int a, b, c;
    printf("Enter in decimal format:");
    scanf("%d", &a);
    printf("Enter in octal format: ");
    scanf("%d", &b);
    printf("Enter in hexadecimal format: ");
    scanf("%d", &c);
    printf("a = %d, b = %d, c = %d", a, b, c);
    printf("Enter in decimal format:");
    scanf("%i", &b);
    printf("Enter in octal format: ");
    scanf("%i", &b);
    printf("Enter in hexadecimal format: ");
    scanf("%i", &c);
    printf("a = %i, b = %i, c = %i\n", a, b, c);
    return 0;}
```

Run the code & Conclude the difference of %d and %i

Specifier	Base Interpretation	Notes
%d	Always decimal	Ignores leading 0 (octal) or 0x (hexadecimal)
%i	Auto-detect	0 → octal, 0x → hex, otherwise decimal