Bismillahhir Rahmanir Rahim

আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম

الجامعة الإسلامية العالمية شيتاغونغ

International Islamic University Chittagong

Assignment

Assignment No. – 10 (Official)

Submission date- 23.07.2020

Course Title-Computer Programming 1 lab

Course Code: CSE- 1121

Submited to-

Mr. Jamil As-ad

Assistant Lecturer, IIUC

Cell: 01626890190

jamilasad1@gmail.com

Submitted by-

MD. SOROWAR MAHABUB RABBY

Matric ID: C201032, Section: A

Department of CSE (Computer Science and Engineering)

1. Explain with appropriate example how the format specifiers mentioned in table 4.1 behaves in printf and scanf function?

Answer :

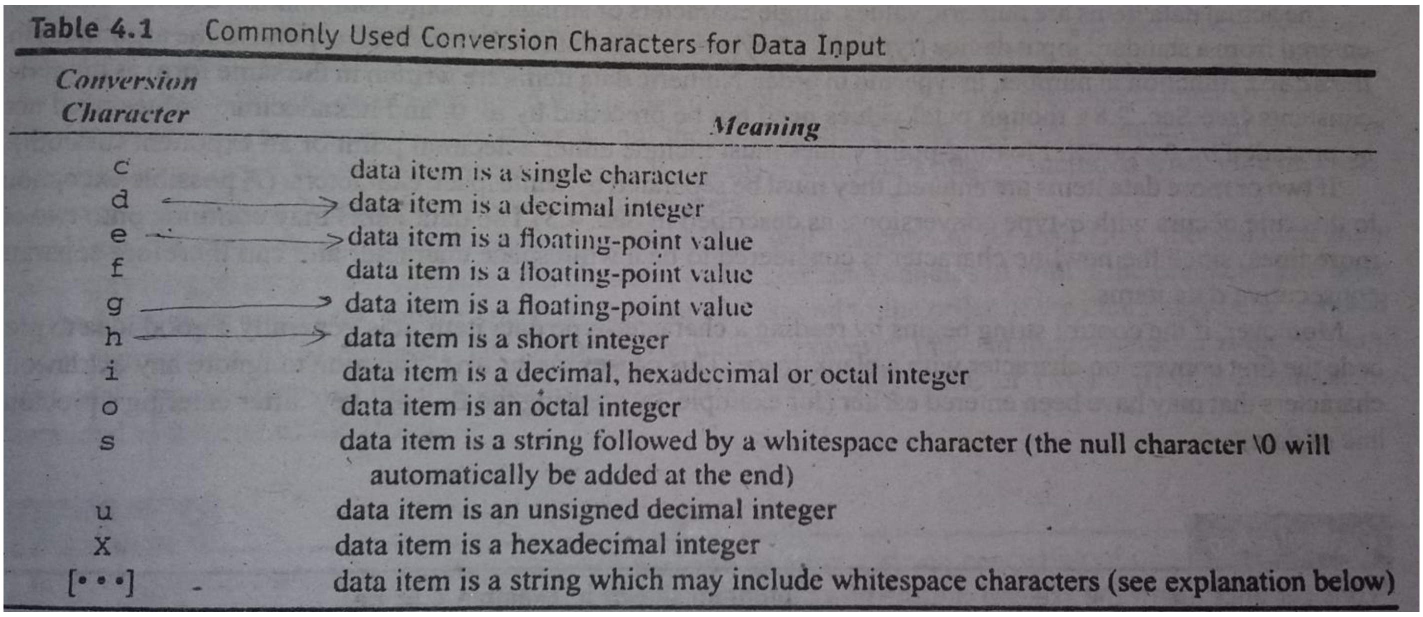


Table 4.1 Commonly Used Conversion Characters for Data Input

Conversion Character	*Meaning*
c	data item is a single character
d	data item is a decimal integer
e	data item is a floating-point value
f	data item is a floating-point value
g	data item is a floating-point value
h	data item is a short integer
i	data item is a decimal, hexadecimal or octal integer
o	data item is an octal integer
s	data item is a string followed by a whitespace character (the null character \0 will automatically be added at the end)
u	data item is an unsigned decimal integer
x	data item is a hexadecimal integer
[• • •]	data item is a string which may include whitespace characters (see explanation below)

Table 4.1 (from book)

Some Format Specifier & their Type-

Format Specifier	Type
%c	Character
%d	Signed integer
%e or %E	Scientific notation of floats
%f	Float values
%g or %G	Similar as %e or %E
%hi	Signed integer (short)
%hu	Unsigned Integer (short)
%i	integer
%l or %ld or %li	Long
%lf	Double
%Lf	Long double
%lu	Unsigned int or unsigned long
%lli or %lld	Long long
%llu	Unsigned long long
%o	Octal representation
%p	Pointer
%s	String

%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Prints nothing
%%	Prints % character
Collected	

Format specifier (character): %c

Ex.- 01

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

```
Input: A
Output: A
```

Submitted by-
MD. SOROWAR MAHABUB RABBY
Matric ID: C201032, **Section**: A
Department of CSE (Computer Science and Engineering)

Ex.- 02

```
#include <stdio.h>
int main()
{
    int data = 65;
    printf("%c\n", data);
    return 0;
}
```

```
Output: A
```

Explanation : %c is used to input & also output a single character .In both examples, we can see **%c** convert data in character and printf function print it on the console, even we input an integer.

Format specifiers (integer): %d, %i, %u

Ex.- 03

```
#include <stdio.h>
int main()
{
   int data;
   scanf("%d", &data);
   printf("%d\n", data);
   printf("%u\n", data);
   printf("%i\n", data);
   return 0;
}
```

```
Input: 65

Output:

65

65

65
```

Ex.- 04

```
#include <stdio.h>
int main()
{
   int data1, data2, data3;
   printf("Enter value in decimal format:");
   scanf("%d",&data1);
   printf("data1 = %i\n\n", data1);
   printf("Enter value in hexadecimal format:");
   scanf("%i",&data2);
   printf("data2 = %i\n\n", data2);
   printf("Enter value in octal format:");
```

```
    scanf("%i",&data3);
    printf("data2 = %i\n\n", data3);
    return 0;
}
```

Explanation : Here, difference between %d and %i format specifier When we are printing using the printf function, there is no specific difference between the %i and %d format specifiers. But both format specifiers behave differently with scanf function. The %d format specifier takes the integer number as decimal but the %i format specifier takes the integer number as decimal, hexadecimal or octal type. it means the %i automatically identified the base of the input integer number.

It is notable that we must put ‘0x’ for hexadecimal number and ‘0’ for octal number while entering the input number.

Format specifiers (float) : %f, %e or %E

Ex.- 05

```
#include <stdio.h>

int main()

{

    float data;

    scanf("%f", &data);

    printf("%f\n", data);

    printf("%e\n", data);

    return 0;

}
```

```
Input: 6.27
Output:
6.270000
6.270000e+000
```

Submitted by-

MD. SOROWAR MAHABUB RABBY

Matric ID: C201032, **Section**: A

Department of CSE (Computer Science and Engineering)

Use of special elements with %f

Ex.- 06

```
#include <stdio.h>

int main()

{

   float data = 6.276240;

   printf("%f\n", data);

   printf("%0.2f\n", data);

   printf("%0.4f\n", data);

   return 0;

}
```

```
Output:

6.276240

6.28

6.2762
```

Explanation : we can see, how we can control the precision of float by placing elements with a format specifier. Here %.2f and %.4f will restrict the values up to two and four decimal values.

Submitted by-
MD. SOROWAR MAHABUB RABBY
Matric ID: C201032, **Section**: A
Department of CSE (Computer Science and Engineering)

Ex.- 07

```
#include <stdio.h>
int main()
{
   int pos = 14;
   float data = 1.2;
   printf("%*f",pos,data);
```

```
    return 0;
}
```

```
The output of the above code will be 1.200000 with 6 space.
```

Explanation : Here 1.200000 is printing with, 6 spaces, because by giving * in printf we can specify an additional width parameter, here ‘pos’ is the width and ‘data’ is the value. if the number is smaller than the width then rest is filled with spaces.

Differences between %f, %e and %g format specifiers

To understand the difference between %f, %e, and %g format specifier we can look for a while bellows Code:

Ex.- 08

```c
#include <stdio.h>
int main(void)
{
    double data1 = 123456.0;
    printf("%e\n", data1);
    printf("%f\n", data1);
    printf("%g\n", data1);
    printf("\n");
    double data2 = 1234567.0;
    printf("%e\n", data2);
    printf("%f\n", data2);
    printf("%g\n", data2);
    return 0;
}
```

```
Output:


1.234560e+005

123456.000000

123456


1.234567e+006

1234567.000000
```

```
1.23457e+006
```

Explanation : When using the G (or g) conversion specifier, the double argument representing a floating-point number is converted in style f or e (or in style F or E), depending on the value converted and the precision.

Format specifiers (octal number): %o

Ex.- 09

```
#include <stdio.h>
int main()
{
   int data = 65;
   printf("%o\n", data);
   return 0;
}
```

```
Output: 101
```

Format specifier (Hexadecimal number): %x, %X

Ex.- 10

```
#include <stdio.h>
int main()
{
   int data = 11;
   printf("%x\n", data);
   return 0;
}
Output: b
```

Format specifier (character array or string): %s

Ex.- 11

```
#include <stdio.h>
int main()
{
   char b [] = "helloworld";
   printf("%s\n", b);
   return 0;
}
```

```
Output: helloworld
```

Use of special elements with %s

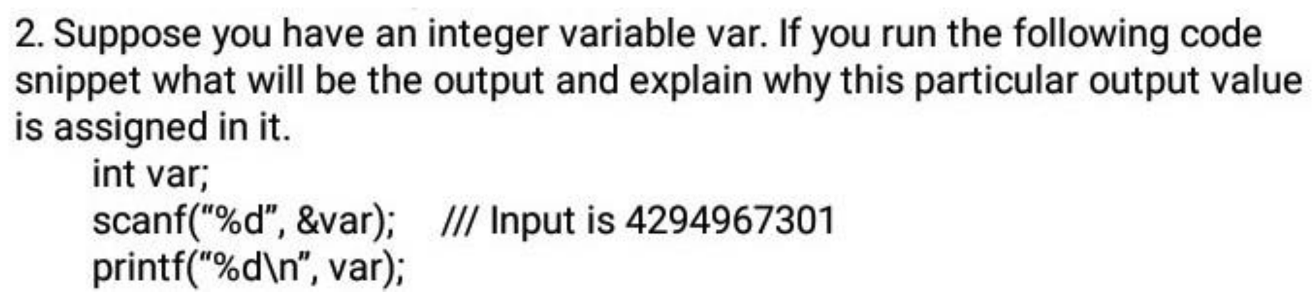
Ex.- 12

```
#include <stdio.h>
int main()
{
   char bl [] = "helloworld";
   printf("%s\n", b);
   printf("%24s\n", b);
   printf("%-24s\n", b);
   printf("%24.6s\n", b);
   printf("%-24.6s\n", b);
   return 0;
}
```

In the mentioned code we can see how – and + is used for left and right alignment. The value after the decimal represents precision.

Note: A precision explains how many digits come after the decimal part in floating number, number of digits in integer, and number of characters in the string.

Submitted by-
MD. SOROWAR MAHABUB RABBY
Matric ID: C201032, **Section**: A
Department of CSE (Computer Science and Engineering)

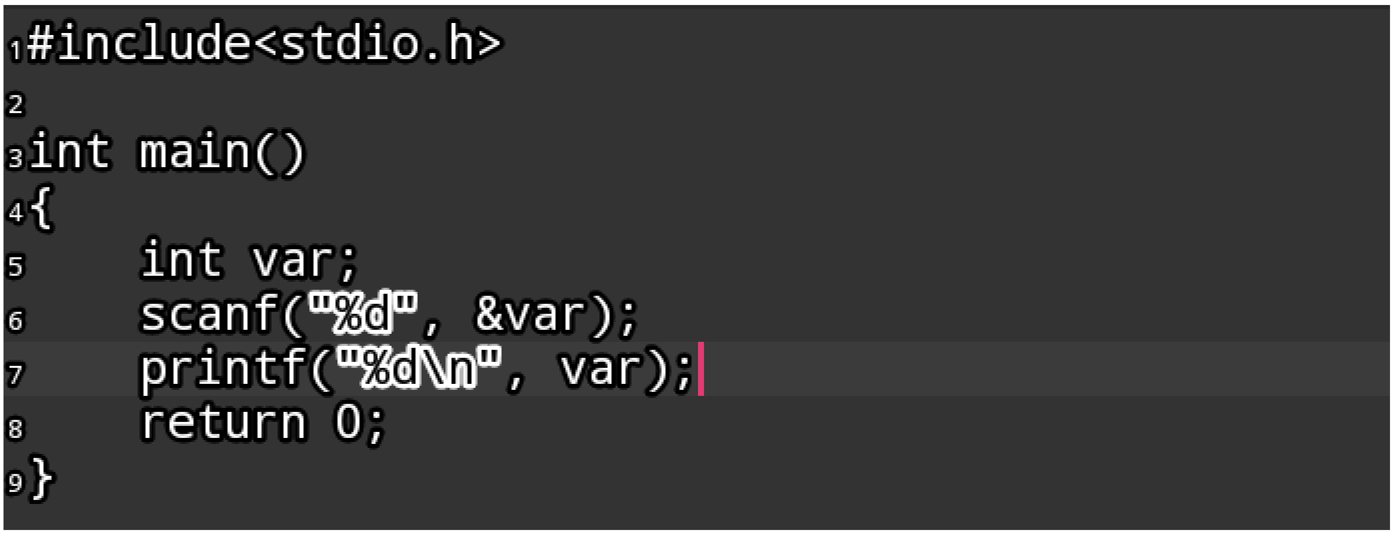


2. Suppose you have an integer variable var. If you run the following code snippet what will be the output and explain why this particular output value is assigned in it.

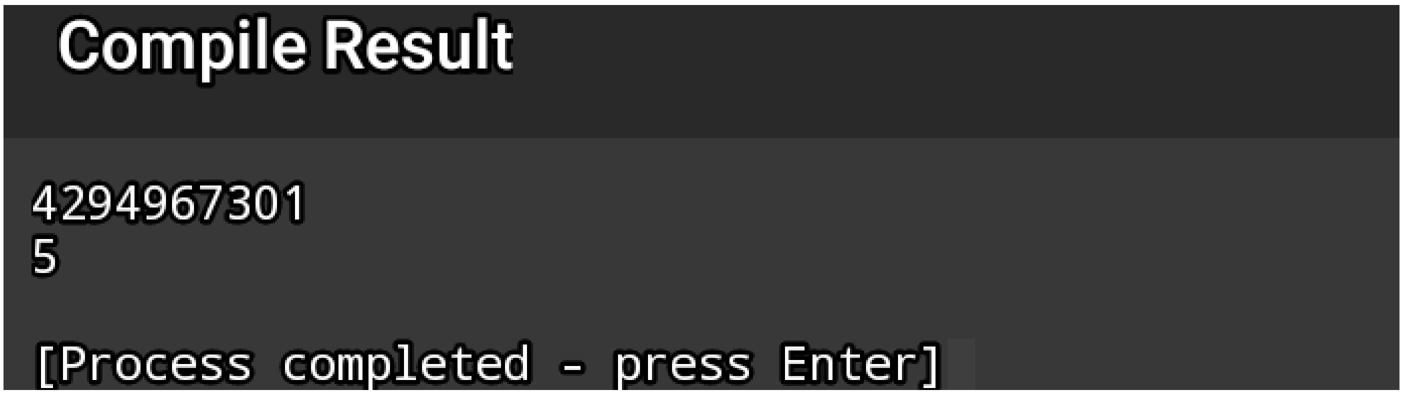
```
int var;
scanf("%d", &var);    /// Input is 4294967301
printf("%d\n", var);
```

Answer :

Firstly, Let’s see the code & compile result.



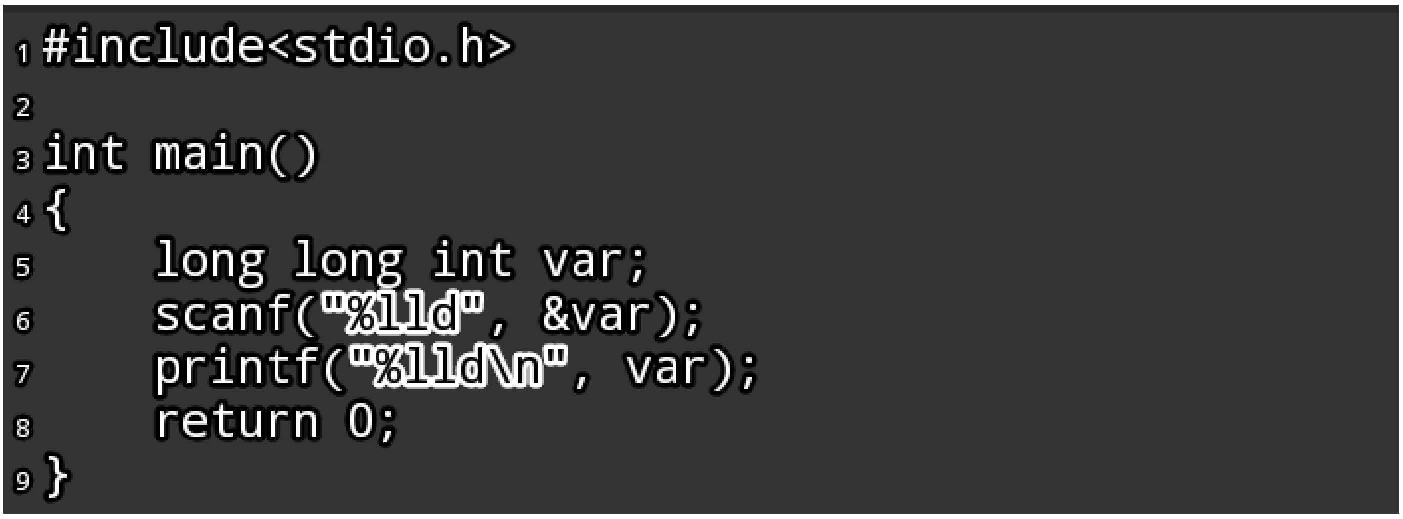
```c
#include<stdio.h>

int main()
{
    int var;
    scanf("%d", &var);
    printf("%d\n", var);
    return 0;
}
```



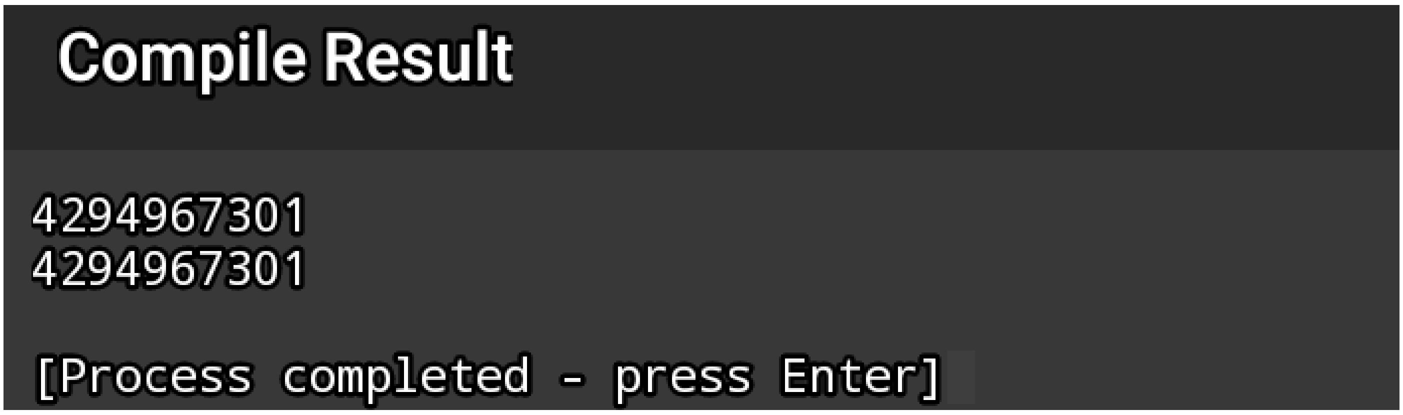
Compile Result

```
4294967301
5

[Process completed - press Enter]
```

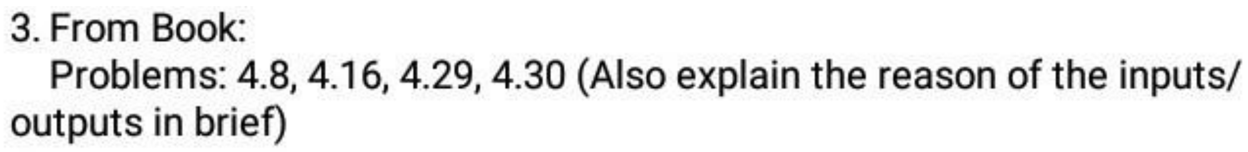
In this above code, when we provide *4294967301* as input the output will be ***5***; it is because (*var*= 4294967301 % 4294967296 , *var*= 5) overflow .When we take the variable *var* as **integer**, the value will be overflowed , & print the value of remainder is ***5***. It can be managed if we use **long long int** replacing the **integer,** then the program returns the exact value (*4294967301* as input) as output. For clarification , let’s a look at the below code:



```c
#include<stdio.h>

int main()
{
    long long int var;
    scanf("%lld", &var);
    printf("%lld\n", var);
    return 0;
}
```


Compile Result

```
4294967301
4294967301

[Process completed - press Enter]
```


3. From Book:
Problems: 4.8, 4.16, 4.29, 4.30 (Also explain the reason of the inputs/ outputs in brief)

Answer of 4.16**:**

a) 12 -8 0.011 -2.2e6

b) 12 -8 0.011 -2.2e6

c) 12 -8 0.011 -2.2e6

d) 12 -8 0.011 -2.2e6

Here,

As in Num (a , b) ,12 will be assigned to a, +8 will be assigned to b, 0.011 will be assigned to x and -2.2e6 will be assigned to y.

field with is specified as in Num(c, d),

%2d--> the first two digits will be assigned to a(1 & 2)

%2d--> the second two digits will be assigned to b(blank space & -)

%5f-->the next 5 digits will be assigned to x (8, blank space, 0, . and 0)

%6e--> the last remaining (/6) digits will be assigned to y (1, 1, blank space, -, 2 and .)

%3d--->the first three digits will be assigned to a(1, 2 and blank space)

%3d--> the second three digits will be assigned to b(-, 8 and black space)

%8f-->the next 8 digits will be assigned to x

%8e--> the last remaining (/6) digits will be assigned to y

Submitted by-
MD. SOROWAR MAHABUB RABBY
Matric ID: C201032, **Section**: A Department of CSE (Computer Science and Engineering)

Answer for 4.8:

a) A long Decimal integer value will be assigned to a with a maximum field-width of 12, A short Decimal integer value will be assigned to b with a maximum field-width of 5, Double precision's value will be assigned to c & d with a maximum field-width of 15.

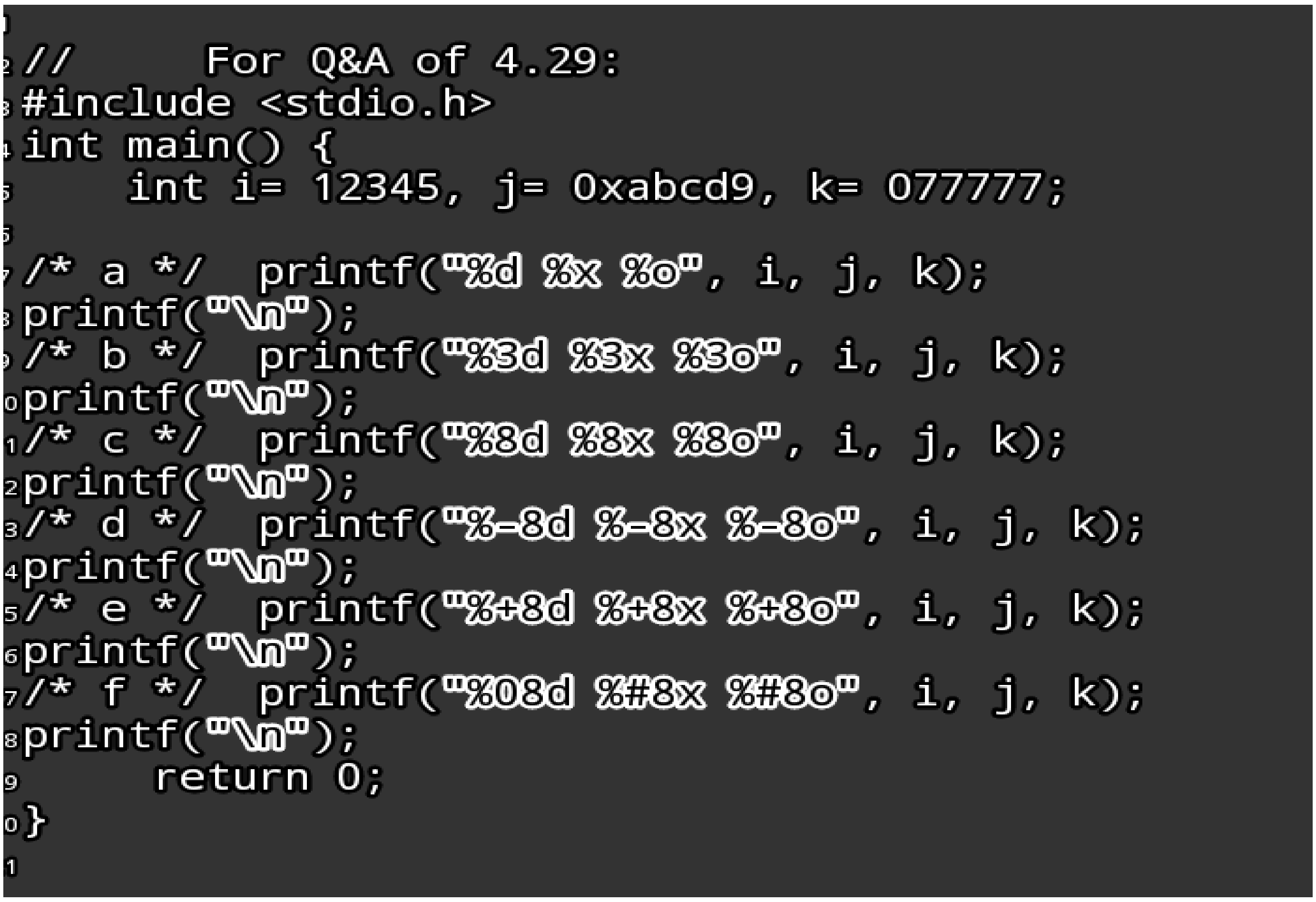
b) A long Hexadecimal integer Will be assigned to a with a maximum field-width of 10, A short octal integer Will be assigned to b with a maximum field-width of 6, A short unsigned integer Will be assigned to c with a maximum field-width of 6 and A long unsigned integer Will be assigned to d with a maximum field-width of 14.

c) A long decimal integer Will be assigned to a with a maximum field-width of 12, A short Decimal integer Will be assigned to b with unspecified field-width, Floating points value will be assigned to c & d with a maximum field-width of 15.

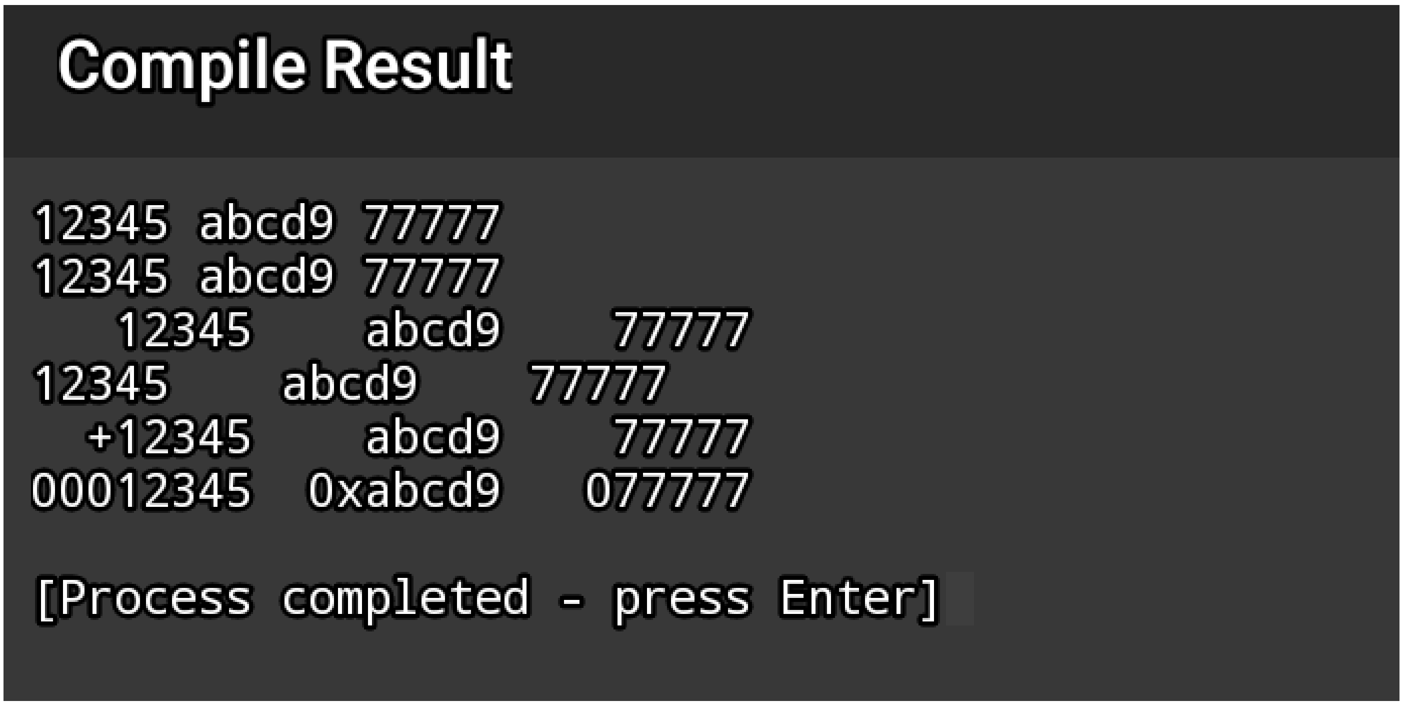
d) A decimal integer Will be assigned to a with a maximum field-width of 8, A Decimal integer Will be read again but assigned to b with specified field-width, Double precision's value will be read & assigned to c & d with a maximum field-width of 12.

Submitted by-
MD. SOROWAR MAHABUB RABBY
Matric ID: C201032, **Section**: A
Department of CSE (Computer Science and Engineering)

Answer :



```c
//      For Q&A of 4.29:
#include <stdio.h>
int main() {
    int i= 12345, j= 0xabcd9, k= 077777;

/* a */  printf("%d %x %o", i, j, k);
printf("\n");
/* b */  printf("%3d %3x %3o", i, j, k);
printf("\n");
/* c */  printf("%8d %8x %8o", i, j, k);
printf("\n");
/* d */  printf("%-8d %-8x %-8o", i, j, k);
printf("\n");
/* e */  printf("%+8d %+8x %+8o", i, j, k);
printf("\n");
/* f */  printf("%08d %#8x %#8o", i, j, k);
printf("\n");
     return 0;
}
```



Compile Result

```
12345 abcd9 77777
12345 abcd9 77777
   12345    abcd9    77777
12345    abcd9    77777
  +12345    abcd9    77777
00012345  0xabcd9   077777

[Process completed - press Enter]
```

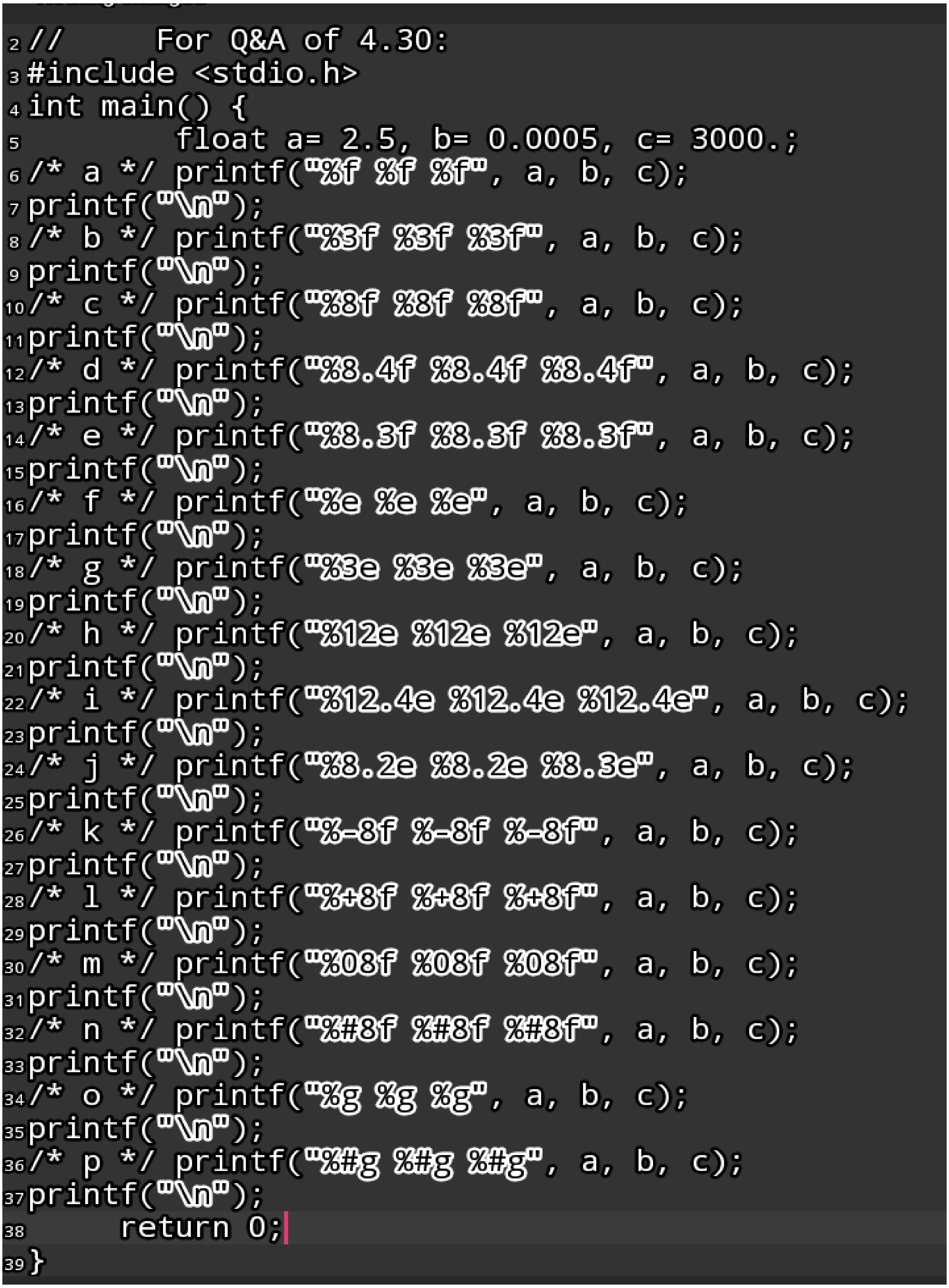
Submitted by-

MD. SOROWAR MAHABUB RABBY

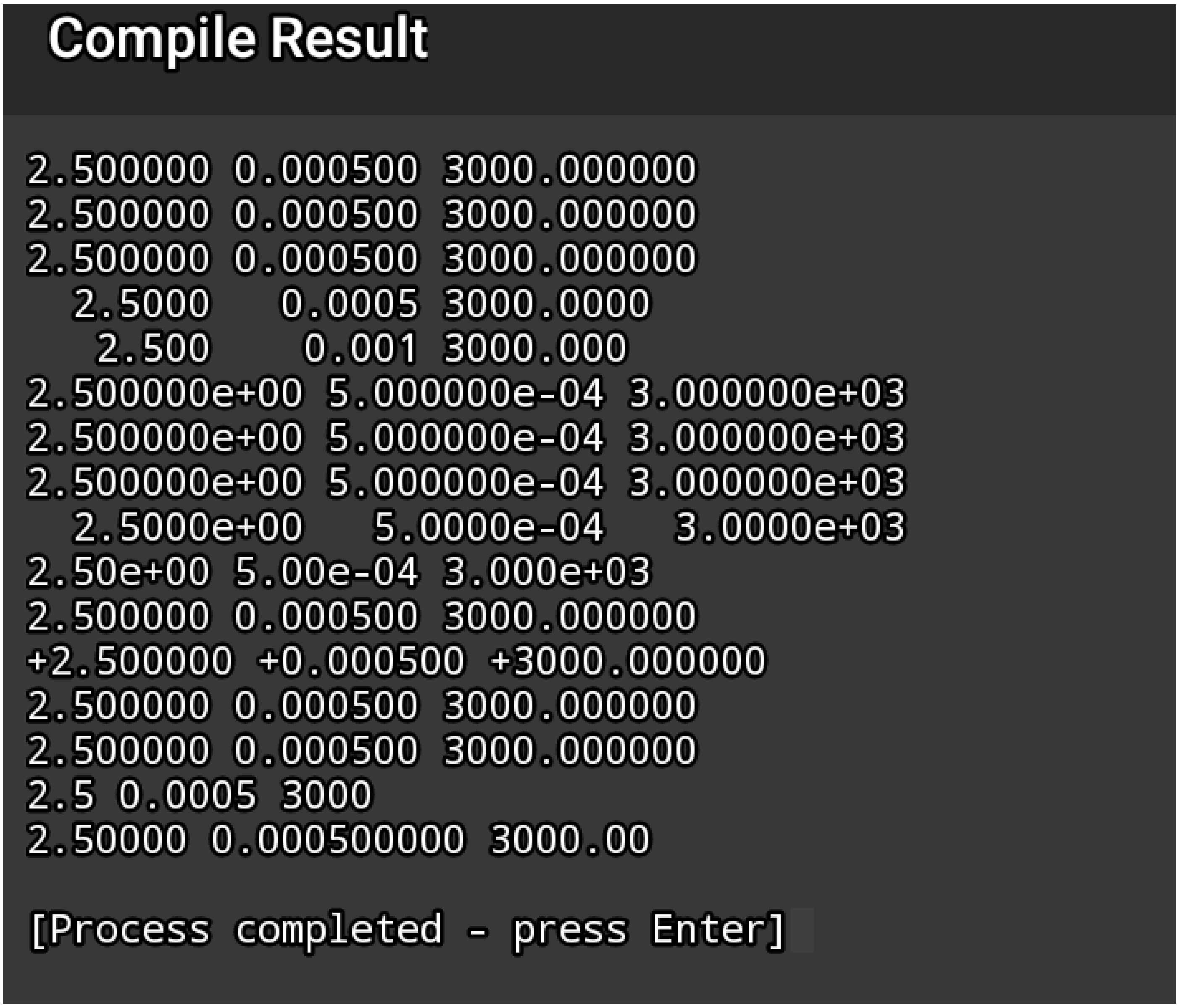
Matric ID: C201032, **Section**: A

Department of CSE (Computer Science and Engineering)

Answer :



```c
//      For Q&A of 4.30:
#include <stdio.h>
int main() {
        float a= 2.5, b= 0.0005, c= 3000.;
/* a */ printf("%f %f %f", a, b, c);
printf("\n");
/* b */ printf("%3f %3f %3f", a, b, c);
printf("\n");
/* c */ printf("%8f %8f %8f", a, b, c);
printf("\n");
/* d */ printf("%8.4f %8.4f %8.4f", a, b, c);
printf("\n");
/* e */ printf("%8.3f %8.3f %8.3f", a, b, c);
printf("\n");
/* f */ printf("%e %e %e", a, b, c);
printf("\n");
/* g */ printf("%3e %3e %3e", a, b, c);
printf("\n");
/* h */ printf("%12e %12e %12e", a, b, c);
printf("\n");
/* i */ printf("%12.4e %12.4e %12.4e", a, b, c);
printf("\n");
/* j */ printf("%8.2e %8.2e %8.3e", a, b, c);
printf("\n");
/* k */ printf("%-8f %-8f %-8f", a, b, c);
printf("\n");
/* l */ printf("%+8f %+8f %+8f", a, b, c);
printf("\n");
/* m */ printf("%08f %08f %08f", a, b, c);
printf("\n");
/* n */ printf("%#8f %#8f %#8f", a, b, c);
printf("\n");
/* o */ printf("%g %g %g", a, b, c);
printf("\n");
/* p */ printf("%#g %#g %#g", a, b, c);
printf("\n");
      return 0;
}
```



```
2.500000 0.000500 3000.000000
2.500000 0.000500 3000.000000
2.500000 0.000500 3000.000000
  2.5000   0.0005 3000.0000
   2.500    0.001 3000.000
2.500000e+00 5.000000e-04 3.000000e+03
2.500000e+00 5.000000e-04 3.000000e+03
2.500000e+00 5.000000e-04 3.000000e+03
  2.5000e+00   5.0000e-04   3.0000e+03
2.50e+00 5.00e-04 3.000e+03
2.500000 0.000500 3000.000000
+2.500000 +0.000500 +3000.000000
2.500000 0.000500 3000.000000
2.500000 0.000500 3000.000000
2.5 0.0005 3000
2.50000 0.000500000 3000.00

[Process completed - press Enter]
```

Here in 4.29 & 4.30 's Code, We can see the uses of flag such as 0, #, +, - and also about the uses of of format specifiers, which are explained avobe.

Submitted by-

MD. SOROWAR MAHABUB RABBY

Matric ID: C201032, **Section**: A

Department of CSE (Computer Science and Engineering)