CONTENTS

Sl No.	Assignment No.	Date	Remarks

Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

WEEK-END ASSIGNMENT-00

Operating Systems Workshop (CSE 3541)

Problem Statement:

Working with general form of C program, formatted output function-**printf()** and formatted input function-**scanf()**

Assignment Objectives:

Familiarization with the general form of a C program, various ways to use printf() to output data items/ messages on standard output device and scanf() function to input of data items from standard input device.

Instruction to Students (If any):

This assignment is designed to deal with general structure of C program, use of printf() function and scanf() function. In this assignment, students are required to write their programs to solve each and every problem as per the specification to meet the basic requirement of systems programming. **Students are required to write the output/ paste the output screen shots onto their laboratory record after each question.**

Programming/ Output Based Questions:

1. Write a C program to display the following messages.

```
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 know:\n");
      printf("\twhere to write C program,\n");
      printf("\thow 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;
}
```

Write/paste output in exact form as expected:

```
This is my first C Program!
I know:
      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 !!!
```

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 from. Mention the two different form int 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
3.000000e+03 3.500000e-03 1.050000e+01
3.000000E+03 3.500000E-03 1.050000E+01
```

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 your Statement:

```
printf("The area of a square whose side length is %f cm is %f square cm.\n",s,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 would you 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.

Items	Data Types	Format Specifier
Number of students in your section	Integer	%d
A letter grade on the AD-1 exam	Character	%c
Average number of days in a semester	Integer	%d
The name of the topper of your class	String	%s
Total number of courses in this semester	Integer	%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;
}
```

Write/paste output in exact form as expected:

```
:54321: :54321: :     54321: :       54321:
:876.543030: :876.543030: :   876.543030: :876.543030:
```

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;
}
```

Write/paste output in exact form as expected:

```
:54321: :54321: :54321     : :       54321:
:876.543030: :876.543030: :876.543030   : :876.543030:
```

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;
}
```

Write/paste output in exact form as expected:

```
:      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;
}
```

Write/paste output in exact form as expected:

```
:      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(":%*.*d:\n",13,7,ivar);
 printf(":%-*.*d:\n",13,7,ivar);
 return 0;
}
```

Write/paste output in exact form as expected:

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

Write/paste output in exact form as expected:

```
: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;
}
```

Write/paste output in exact form as expected:

```
123.456001 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;
}
```

Write/paste output in exact form as expected:

```
: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;
}
```

Write/paste output in exact form as expected:

```
: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;
 }
```

Write/paste output in exact form as expected:

```
:   345: :     34: :  -5.6e+00:
:345   : :34      : :-5.6e+00  :
:  +345: :    +34: :  -5.6e+00:
:+345  : :+34     : :-5.6e+00  :
:21688992: :    34.: :10g: :   34.0000:
```

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;
}
```

Write/paste output in exact form as expected:

```
:     345: :    1767: :    a0bd:
:345     : :1767    : :a0bd    :
:     345: :   01767: :  0xa0bd:
:00000345: :17670: :0000a0bd:
:     345: :   01767: :  0xa0bd:
```

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;
}
```

Write/paste output in exact form as expected:

```
:     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;
}
```

Write/paste output in exact form as expected:

```
$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 char1, char2, char3;
      printf("Enter three characters : ");
      scanf(" %c %c %c", &char1, &char2, &char3);
      printf("Characters entered: %c %c %c\n", char1, char2, char3);
      return 0;
}
```

```
#include <stdio.h>
int main() {
      char input[4];
      printf("Enter three characters separated by spaces: ");
      scanf("%s", input);
      printf("Characters entered: %c %c %c\n", input[0], input[1],
      input[2]);
      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
(B) 100
(C) 10
(D) Error

Answer with reason:

100
Reason:
The second argument m is not used in the printf statement.
(Two placeholders are required)

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
(B) 0 100
(C) 0 16
(D) 16 0
(E) Compilation error
(F) Run-time error

Answer with reason:
100

Reason:
scanf returns 0
printf returns ::16
sr = 0 because there are no format specifiers to match the input.
pr = 0 because the number of characters printed in the inner printf statement is 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

(B) 23 9 76 45

(C) 456 76 9 2345

(D) No output

Answer with reason:
23 9 76 45

Reason:
'%2d' in scanf means it takes only two literals from the input 'num', So, in first input case of '2345', only the starting two literals are taken as input i.e., '23'.

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 run.

(A) 2345 9 76 456

(B) 23 9 76 45

(C) 456 76 9 2345

(D) No output

Answer with reason:

Reason:

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

(B) 345678

(C) 726

(D) No output

Answer with reason:

```
Enter the number as <345678>:345678
726
```

Reason:

```
Num1 = 3, num2 = 45, num3 = 678
Therefore, sum = num1 + num2 + 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

(B) 10 10 10

(C) 6 6 6

(D) No output

Answer with reason:

```
10 10 10
```

Reason:

The inner printf returns the number of characters printed, which in this case is 6('1', '0', ' ', '1', '0', ' ').

The outer printf("%d", ……) then prints this return value, which is 6.

29. 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):

```
#include <stdio.h>
int main() {
     int i, j, k;
     printf("Enter decimal values for i, j, and k: ");
     scanf("%d %d %d" , &i, &j, &k);
     printf("i: %d, j: %d , k: %d\n", i, j, k);
     return 0;
}
```

Output:

```
Enter decimal values for i, j, and k: 12 34 56
i: 12, j: 34, k: 56
```

State the output in exact form:

Code for (ii):

```
#include <stdio.h>
int main() {
      int i, j, k;
      printf("Enter decimal, octal, and hexadecimal values for i, j, and
      k: ");
      scanf("%d %o %x", &i, &j, &k);
      printf("i: %d, j: %o, k: %x\n", i, j, k);
      return 0;
}
```

Output:

```
Enter decimal, octal, and hexadecimal values for i, j, and k: 20 075 0x1A
i: 20, j: 075, k: 1a
```

State the output in exact form:

Code for (iii):

```
#include <stdio.h>
int main() {
      int i, j, k;
      printf("Enter hexadecimal values for i and j, and an octal value for
      k: ");
      scanf("%x %x %o", &i, &j, &k);
      printf("i: %x, j: %x, k: %o\n", i, j, k);
      return 0;
}
```

Output:

```
Enter hexadecimal values for i and j, and an octal value for k: 1F 2A 45
i: 1f, j: 2a, k: 45
```

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;
 }
```

State the output in exact form:

```
Enter in decimal format:45
Enter in octal format: 24
Enter in hexadecimal format: A8
a = 45, b = 24, c = 32767Enter in decimal format:Enter in octal format:
Enter in hexadecimal format: a = 45, b = 24, c = 32767
```

Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

WEEK-END ASSIGNMENT-01

Operating Systems Workshop (CSE 3541)

Problem Statement:

Experiment with C operators, role of operator precedence, associativity and expressions.

Assignment Objectives:

To become familiar with C operators, expression evaluation as per operator precedence and associativity rule.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realise the coutcome of that question in relevant to systems programming.

Programming/ Output Based Questions:

1. Evaluate the arithmetic expression *a – b/c* d*, where the floating-point variables a , b , c and d have been assigned the values 1.0, 2.0, 3.0 and 4.0. Create a C program to display the value of the expression on standard output device.

Write/paste your code here	Output

`#include <stdio.h>`
`int main(){`
`  float a = 1.0;`
`  float b = 2.0;`
`  float c = 3.0;`
`  float d = 4.0;`
`  float res = a - b / c + d;`
`  printf("Result: %f\n",res);`
`  return 0;`
`}`
	`Result: 4.333333`

2. Which of the following identifiers are (a) C reserved words, (b) standard identifiers, (c) conventionally used as constant macro names, (d) other valid identifiers, and (e) invalid identifiers?

`void`	`MAX_ENTRIES`	`return`	`printf`	`"char"`
`xyz123`	`time`	`part#2`	`G`	`Sue's`
`#insert`	`this_is_a_long_one`		`double`	`__hello_`

3. The Pythagorean theorem states that the sum of the squares of the sides of a right triangle is equal to the square of the hypotenuse. For example, if two sides of a right triangle have lengths of 3 and 4, then the hypotenuse must have a length of 5. Together the integers 3, 4, and 5 form a *Pythagorean triple*. There are an infinite number of such triples. Given two positive integers, m and n, where $m > n$, a Pythagorean triple can be generated by the following formulas:

$$side1 = m^2 - n^2$$

$$side2 = 2mn$$

$$hypotenuse = m^2 + n^2$$

The triple ($side1 = 3$, $side2 = 4$, $hypotenuse = 5$) is generated by this formula when $m = 2$ and $n = 1$. Write a program that takes values for m and n as input and displays the values of the Pythagorean triple generated by the formulas above. The values of m and n should be provided from an input file through input redirection.

Write/paste your code here	**Output**

#include <stdio.h>
#include <math.h>
int main(){
int m, n;
printf("Enter m: ");
scanf("%d",&m);
printf("Enter n: ");
scanf("%d",&n);
float side1=pow(m, 2)-pow(n,2);
float side2=2*m*n;
float hypotenuse=pow(m,2)+pow(n,2);
printf("The triple is:(side1=%g, side2=%g, hypotenuse=%g)\n",side1,side2,hypotenuse);
return 0;
}
	Enter m: 2
	Enter n: 1
	The triple is:(side1=3, side2=4, hypotenuse=5)

4. Write C statements to carry out the following steps.

   a. If **item** is nonzero, then multiply **product** by **item** and save the result in **product** ; otherwise, skip the multiplication. In either case, print the value of **product**.

   b. Store the absolute difference of **x** and **y** in **y** , where the absolute difference is **(x - y)** or **(y - x)**, whichever is positive. Do not use the **abs** or **fabs** function in your solution.

   c. If **x** is 0 , add 1 to **zero_count**. If **x** is negative, add **x** to **minus_sum**. If **x** is greater than 0 , add **x** to **plus_sum**.

Write/paste your code here

a)

```
#include <stdio.h>
int main(){
  int item,product;
  printf("Enter
item");
  scanf("%d",&item);
  printf("Enter
product");
  scanf("%d",&product);
  if(item==0){
        printf("Invali
d - Item cannot be
zero\n");
        return 0;
    }else{
        product =
product * item;
    }
 printf("Product:
%d\n",product);
    return 0;
}
```

b)

```
#include <stdio.h>
int main(){
    int x,y;
    printf("Enter x: ");
    scanf("%d",&x);
    printf("Enter y: ");
    scanf("%d",&y);
    if(x-y>0){
        y = x-y;
    }else{
        y = y-x;
    }
    printf("y = %d\n",y);
    return 0;
}
```

c)

```
#include <stdio.h>
int main(){
    int data,x;
    printf("Enter no. of
data: ");
    scanf("%d",&data);
    int zero_count = 0;
    int minus_sum = 0;
    int plus_sum = 0;
    printf("Enter x: ");
    scanf("%d",&x);
    if(x>0){
        plus_sum += x;
    }else if(x<0){
        minus_sum += x;
    }else{
        zero_count++;
    }
    printf("minus_sum =
%d\n",minus_sum);
    printf("plus_sum =
%d\n",minus_sum);
    printf("zero_count =
%d\n",minus_sum);
    return 0;
}
```

5. Consider the C arithmetic expression 2 * ((i%5) * (4+(j - 3) / (k +2))) where i, j and k are integer variables. If these variables are assigned the values 8, 15 and 4, respectively, then the given determine the value of the expression. (Note: The interpretation of the remainder operation (%) is unclear when one of the operands is negative. Most versions of C assign the sign of the first operand to the remainder. The % operation is undefined when second operand is zero.)

Expression Evaluation:
1. Evaluate the innermost parenthesis:
4+(j - 3) / (k +2) = 4 + (15 - 3)/(4 + 2) = 4 + 12/6 = 6.
2. Evaluate parenthesis around the modulus operator:
(i%5) * 6 = (8%5) * 6 = 3*6 =18.
3. Multiply the two results:
2 * 18 = **36 (OUTPUT)**

6. Consider the following C expressions;

   a. Suppose that *i* is an integer variable whose value is 7, and *f* is a floating-point variable whose value is 8.5. The expression (*i* + *f*)%4 is valid or invalid.

   b. Suppose that *i* is an integer variable whose value is 7, and *f* is a floating-point variable whose value is 8.5. The expression ((*int*)(*i* + *f*))%4 is valid or invalid.

Answer & Cause
a) *Invalid*: The modulus operator (%) can only be used with integer operands. In this case, the variable f is a floating-point variable, so the expression is invalid.

b) *Valid*: It is evaluated from left to right, with parentheses being evaluated first. The expression first casts i + f to an integer, and then takes the modulus of the result. If i is 7 and f is 8.5, the expression **evaluates to 3**.

7. ASCII code for the character ? is 63. Characters are represented by integer codes, C permits conversion of type **`char`** to type **`int`** and vice versa. So find the output for the given code snippet;

```
int q_code = (int)'?';
printf("%d %c %d\n", q_code,'?','?');
```

Output:
63 ? 63

8. The following expressions contain different operands and operators assuming **x=3.0** , **y=4.0** , and **z=2.0** are type **double** , **flag=0** is type **int**. Write each expressions value.

(a) ! flag
(b) x + y/z <= 3.5
(c) !flag || (y + z >= x - z)
(d) !(flag || (y + z >= x - z))

Answer:
(a) ! flag : **1**
(b) x + y / z <= 3.5 : **0**
(c) !flag || (y + z >= x - z): **true**
(d) !(flag || (y + z >= x - z)): **false**

9. What value is assigned to the type **`int`** variable **`ans`** in this statement if the value of **`p`** is **`100`** and **`q`** is **`50`**?

```
ans = (p > 95) + (q < 95);
```

Output:
ans = 2

10. Evaluate each of the following expressions if `a` is 6 , `b` is 9 , `c` is 14 , and flag is 1 . Which parts of these expressions are not evaluated due to short-circuit evaluation?

(a) `c == a + b || !flag`

(b) `a != 7 && flag || c >= 6`

(c) `!(b <= 12) && a % 2 == 0`

(d) `!(a > 5 || c < a + b)`

Answer:
(a) ! flag : **1**
(b) x + y / z <= 3.5 : **0**
(c) !flag || (y + z >= x - z): **true**
(d) !(flag || (y + z >= x - z)): **false**

11. Suppose that **`i`** is an integer variable, **`x`** is a floating-point variable, **`d`** is a double-precision variable and **`c`** is a character-type variable. Find the output generated by these statements that make use of the operator **`sizeof`**.

```
printf("integer:%ld bytes\n", sizeof i);
printf("integer:%ld bytes\n", sizeof(i));
printf("float:%ld bytes\n", sizeof x);
printf("float:%ld bytes\n", sizeof(x));
printf("double:%ld bytes\n", sizeof d);
printf("double:%ld bytes\n", sizeof(d));
printf("character:%ld bytes\n", sizeof c);
printf("character:%ld bytes\n", sizeof(c));
/* Same way can be used for other data types to
     find the size */
```

Answer:
```
integer:4 bytes
integer:4 bytes
float:4 bytes
float:4 bytes
double:8 bytes
double:8 bytes
character:1 bytes
character:1 bytes
```

12. Another way to generate the same information as like previous question is to use a cast rather than a variable within each printf statement. Find the output generated by these statements that make use ofthe operator **`sizeof`**.

```
printf("integer:%ld bytes\n", sizeof(int));
printf("float:%ld bytes\n", sizeof(float));
printf("double:%ld bytes\n", sizeof(double));
printf("character:%ld bytes\n", sizeof(char));
/* Same way can be used for other data types to
     find the size */
```

Answer:
```
integer:4 bytes
float:4 bytes
double:8 bytes
character:1 bytes
```

13. C supports several assignment operators. The most commonly used assignment operator is =. Assignment expressions that make use of this operator are written in the form **`identifier = expression`**, where i**`dentifier`** generally represents a variable, and **`expression`** represents a constant, a variable or a more complex expression. Determine the expression values assume that i is an integer-type variable, and that the ASCII character set applies.

```
i=('x'-'o')/3;
i=('y'-'o')/3;
i=2*j/2; (say j is an integer and j is 5)
i=2*(j/2);
i=3.0;
i=-3.5;
```

Answer:
```
i = ('x'-'o')/3 => i = 3
i = ('y'-'o')/3 => i = 3
i = 2*j/2       => i = 5
i = 2*(j/2)     => i = 5
i = 3.0         => i = 3
i = -3.5        => i = -3
```

NOTE: Multiple assignments of the form
`identifier 1 = identifier 2 = ...  = expression`
are permissible in C. In such situations, the assignments are carried out from **right to left**.

14. C also contains other form assignment operators: +=, -+, *=, /=, %= etc., called short hand operators. Suppose that i and j are integer variables whose values are 5 and 7, and f and g are floating-point variables whose values are 5.5 and -3.25. Determine the value of the expressions

```
i += 5;
f -= g;
j *= (i - 3);
f /= 3;
i %= (j - 2);
```

Answer:

```
i += 5      => i = 10
f -= g      => f = 8.750000
j *= (i-3)  => j = 14
f /= 3      => f = -278774856
i %= (j-2)  => i = 0
```

15. Suppose that x, y and z are integer variables which have been assigned the values 2, 3 and 4, respectively. Determine the value of the given expression;

```
x*=-2*(y+z)/3;
```

Answer:

```
x = -8 c
```

16. The assignment statement that contains a conditional expression on the right-hand side. Determine the value of flag if **i=-5** and **i=-6** respectively.

```
flag = ( i < 0 ) ? 0 : 100
```

Answer:

```
For i=-5, flag=0
For i=-6, flag=0
```

17. In the following assignment statement, a, b and c are assumed to be integer variables. If **a**, **b** and **c** have the values 1, 2 and 3, respectively, then determine the value of the expression that includes operators of different precedence groups.

```
c += (a > 0 && a <= 10) ? ++a : a / b ;
```

Answer:

```
c = 5
```

18. Illustrate the purpose of the following code snippet over the inputs a, b and c respectively.

```
int m1,m2,a,b,c;
printf("Enter the values of a,b,c:");
scanf("%d%d%d",&a,&b,&c);
m1=(a>b)?a:b;
m2=(m1>c)?m1:c;
printf("%d\n",m2);
```

Answer:
The purpose of the following code snippet is to find the maximum or largest integer among the three.

19. A C program contains the following declarations and initial assignments:

```
int i= 8;
int j = 5;
float x = 0.005;
float y = -0.01;
char c = 'c' , d = 'd ' ;
```

Determine the value of each of the following expressions. Use the values initially assigned to the variables for each expression.

```
(a) ( 3 * i - 2 *j ) % ( 2 * d - c )
(b) ( x > y ) && (i > 0) && ( j < 5 )
(c) 2 * x + ( y = = 0)
(d) ( 2 * x + y ) == 0
(e) 5 * (i + j ) > ' c '
(f) i++
```

Answer:
(a) 14
(b) 0
(c) 0
(d) 1
(e) 0
(f) 9

20. Suppose a is an unsigned integer variable (say represented in 16 bits format) whose value is 0x6db7. In the following the expression, we will shift all bits of a six places to the right and assign the resulting bit pattern to the unsigned integer variable b. Find the resulting value of b. Also write the lost bits because of shifting.

```
b = a >> 6 ;
```

Answer

21. Determine the value of each of the following expressions, assume that a is an unsigned integer variable whose initial value is 0x6db7.

```
 a. a &= 0x7f
 b. a ^= 0x7f
 c. a |= 0x7f
 d. a = a & 0x3f06
(e) a = a | 0x3f06 << 8
```

Answer

22. Determine the output of the following code snippet.

```
int main(){
 int m1,a,b,c;
 printf("Enter the values of a,b,c:");
 scanf("%d%d%d",&a,&b,&c);
 m1=a>b?a>c?a:c:b;
 printf("m1=%d\n",m1);
 return 0;
}
```

Answer:

(1) a=10 b=20 c=30 m1=20

(2) a=30 b=10 c=20 m1=30

(3) a=20 b=30 c=10 m1=20

23. Evaluate the expressions:

```
Assume A, B, num, xy, f, t, p, q, r
are int type variables;
(1) A=10+(num=2)*3;
(2) B +=(xy *=3); [here xy=10]
(3) x +=(f=(t*=20)); [here x=20, t=10]
(4) p=q=r=100;
```

Answer:

(1) A = 16
(2) B = 30
(3) x = 1728628944
(4) p = 100, q = 100, r = 100

24. State the output of the following code snippet;

```
int a,b,s;
s=scanf("%d%d%d",&a,&b,&a);
printf("%d\n",s+printf("OSW CSE="));
```

Answer:
Output:
```
OSW CSE = 11
```

25. State the output of the following code snippet;

```
int i=-1,j=-1,k=0,l=2,m;
m=++i || k++ && ++j || l++;
printf("%d %d %d %d %d\n", i,j,k,l,m);
```

Answer:
Output:
```
0 -1 1 3 1
```

26. State the output of the following code snippet;

```
int i=10,j=6;
printf("%d\n", i+++j++);
```

Answer:
Output:
```
16
```

27. State the output of the following code snippet;

```
int i=3>4, j=4>3;
int k=(i=j);
int l=(k==j);
printf("%d %d %d %d",i,j,k,l);
```

Answer:
Output:
```
16
```

28. State the output of the following code snippet;

```
int x=400;
printf("%d %d\n",x=40,x>=50);
```

Answer:
Output:
```
40 1
```

29. verify the output/ error of the following code snippet;

```
int i=2,j=0;
int k=i&&j=1;
printf("%d\n",k);
```

Answer:
Error:
```
error: lvalue required as left
operand of assignment
int k = i&&j = 1;
```

30. Find the output of the following code snippet;

```
int i=2,j=2;
int k=i^j&i;
printf("%d\n",k);
```

Answer:
Error:
```
error: 'i\U000002c6j' undeclared
(first use in this function)
int k=i^j&i;
```

31. Find the output of the following code snippet;

```
int i=3,j=2;
int k=i << 1 > 5;
printf("%d\n",k);
```

Answer:
Output:
```
1
```

WEEK-END ASSIGNMENT-02

Operating Systems Workshop (CSE 3541)

Problem Statement:

Experiment with selection structures; if, if-else, if-else if-else and switch statements to develop applications.

Assignment Objectives:

To become familiar with one of the control structures, selection, out of sequence, selection, and repetition kinds of control structure.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realize the outcome of that question in relevant to systems programming. You may use additional pages on requirement.

Programming/ Output Based Questions:

1. Find the value that is assigned to x when y is 10.0.

Code snippet: 1(a)

```
x = 25.0;
if(y != (x - 10.0))
    x = x - 10.0;
else
    x = x / 2.0;
```

Output:

```
15.000000
```

Code snippet: 1(b)

```
if(y < 15.0)
   if(y >= 0.0)
      x = 5 * y;
   else
      x = 2 * y;
else
   x = 3 * y;
```

Output:

```
50.000000
```

Code snippet: 1(c)

```
if (y < 15.0 && y >= 0.0)
   x = 5 * y;
else
   x = 2 * y;
```

Output:

```
50.000000
```

2. Write C statements to carry out the following:

   If item is nonzero, then multiply product by item and save the result in product ; otherwise, skip the multiplication. In either case, print the value of product. Declare the appropriate type and initialize, if required.

Output:

```
Enter item: 5
Enter product: 3
Product : 15
```

3. Correct the following if statement; assume the indentation is correct.

```
if (deduct < balance);
    balance = balance - deduct;
    printf("New balance is %.2f\n", balance);
 else;
    printf("Deduction of %.2f refused.\n",
        deduct);
    printf("Would overdraw account.\n");

printf("Deduction = %.2f Final balance = %.2f",
 deduct, balance);
```

Corrected Code:

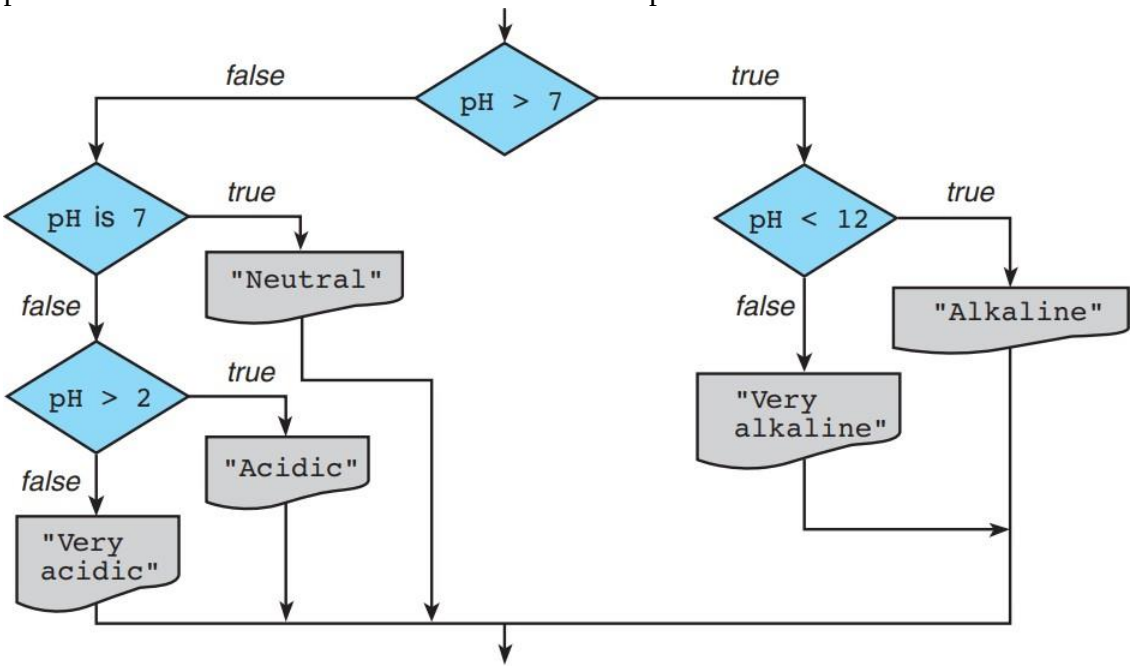
```
if (deduct < balance){
        balance = balance - deduct;
        printf("New balance is %.2f\n", balance);
    }
    else{
        printf("Deduction of %.2f refused.\n",
        deduct);
        printf("Would overdraw account.\n");
        printf("Deduction = %.2f Final balance =
%.2f",deduct, balance);
```

4. Write an interactive program that contains an if statement that may be used to compute the area of a square ($area = side^2$) or a circle ($area = \pi \times radius^2$) after prompting the user to type the first character of the figure name (S or C).

Write/Paste your code here:

```
#include <stdio.h>
#define PI 3.14159265358979323846
int main() {
  char figure;
  float side, radius, area;
  printf("Enter the first character of the figure name (S or C): ");
  scanf("%c", & figure);
  if (figure == 'S') {
    printf("Enter the side of the square: ");
    scanf("%f", & side);
    area = side * side;
  } else if (figure == 'C') {
    printf("Enter the radius of the circle: ");
    scanf("%f", & radius);
    area = PI * radius * radius;
  } else {
    printf("Invalid figure name.\n");
    return 0;
  }
  printf("The area of the figure is %f.\n", area);
  return 0;
}
```

5. Write a nested if statement for the decision diagrammed in the accompanying flowchart. Use a multiple-alternative if for intermediate decisions where possible.



Space for Program and Output:

Program:

```
#include <stdio.h>

#define PI 3.141592
int main() {
  int ph, ph1, ph2, ph3;
  printf("Enter the ph value\n");
  scanf("%d", & ph);
  if (ph > 7)
    if (ph < 12)
      printf("Alkaline\n");
    else
      printf("Very Alkaline\n");
  else
  if (ph == 7)
    printf("Neutral\n");
  else
  if (ph > 2 && ph < 7)
    printf("Acidic\n");
  else
    printf("Very Acidic\n");
  return 0;
}
```

Output:

```
Enter the ph value
6
Acidic
```

6. Implement the following decision table using a **nested if** statement. Assume that the grade point average is within the range 0.0 through 4.0.

Grade Point Average	Transcript Message
0.00.99	Failed semester registration suspended
1.01.99	On probation for next semester
2.02.99	(no message)
3.03.49	Deans list for semester
3.54.00	Highest honors for semester

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
  float gpa;
  printf("Enter your GPA: ");
  scanf("%f", & gpa);
  if (gpa < 0.0 || gpa > 4.0) {
    printf("Invalid GPA.\n");
    return 1;
  }
  if (gpa < 1.0) {
    printf("Failed semester; registration
suspended.\n");
  } else if (gpa < 2.0) {
    printf("On probation for next semester.\n");
  } else if (gpa < 3.0) {} else if (gpa < 3.5) {
    printf("Dean's list for semester.\n");
  } else {
    printf("Highest honors for semester.\n");
  }
  return 0;
}
```

Output:

```
Enter your GPA: 0
Failed semester;
registration suspended

Enter your GPA: 1.25
On probation for next
semester.

Enter your GPA: 2.3
Enter your GPA: 3.02
Dean's list for
semester.

Enter your GPA: 3.52
Highest honors for
semester.
```

7. Implement the following decision table using a **multiple-alternative if** statement. Assume that the wind speed is given as an integer.

Wind Speed (mph)	Category
below 25	not a strong wind
2538	strong wind
3954	gale
5572	whole gale
above 72	hurricane

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
  int wind_speed;
  printf("Enter wind speed (mph): ");
  scanf("%d", & wind_speed);
  if (wind_speed < 25) {
    printf("Category: not a strong wind\n");
  } else if (wind_speed >= 25 && wind_speed <= 38)
{
    printf("Category: strong wind\n");
  } else if (wind_speed >= 39 && wind_speed <= 54)
{
    printf("Category: gale\n");
  } else if (wind_speed >= 55 && wind_speed <= 72)
{
    printf("Category: whole gale\n");
  } else {
    printf("Category: hurricane\n");
  }
  return 0;
}
```

Output:

```
Enter wind speed (mph): 20
Category: not a strong wind

Enter wind speed (mph): 28
Category: strong wind

Enter wind speed (mph): 44
Category: gale

Enter wind speed (mph): 69
Category: whole gale

Enter wind speed (mph): 82
Category: hurricane
```

8. What will be printed by this carelessly constructed switch statement if the value of **color** is **'R'**?

```
switch (color) { /* break statements missing */
case 'R':
   printf("red\n");
case 'B':
   printf("blue\n");
case 'Y':
  printf("yellow\n");
}
```

Output:

```
red
blue
yellow
```

9. Determine life expectancy of a standard light bulb given the input watts; 35, 45, 76, 120 respectively.

```
switch (watts) {
 case 25:
    life = 2500;
    break;
 case 40:
 case 60:
    life = 1000;
    break;
 case 75:
 case 100:
    life = 750;
    break;
 default:
    life = 0;
}
```

Output:

No output

It will not print anything as the print statement is not mentioned as well as the given input watts does not match any of the cases

10. C relational and equality operators give a result of 1 for true and 0 for false. Evaluate the following expression for different values of **x**. Also write the statement to avoid such common error.

```
if (0 <= x <= 4)
    printf("Condition is true\n");
```

Common Error Correction:

```
if(x>=0 && x<=4){
  printf("Condition is true\n");
}
```

Test cases and Output:

```
(a)x=5
(b)x=15
(c)x=34
(d)x=-20
(e)x=-45

No output for any of the
above test cases because it
does not match the condition.
```

11. Evaluate the following code snippet for different values of **x**.

```
printf("Enter x \n");
scanf("%d",&x);
if (x = 10)
 printf("x is 10");
 printf("Differentiate: == and =");
else
 printf(" simply incorrect results");
```

Common Error Correction:

For every test case it will always print the if condition as in the if statement x=10 so any value of x is assigned with 10 so it will only print if statement condition not the else ones.

Correction:

if(x==10)

Test cases and Output:

```
(a)x=5
 x is 10

(b)x=15
 x is 10

(c)x=34
 x is 10

(d)x=-20
 x is 10

(e)x=-45
 x is 10
```

12. Write a switch statement that assigns to the variable lumens the expected brightness of a standard light bulb whose wattage has been stored in watts. Assign 1 to **lumens** if the value of **watts** is not inthe table.Use this table:

Watts	Brightness (in Lumens)
15	125
25	215
40	500
60	880
75	1000
100	1675

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
  int watts, lumens;
  printf("Enter the wattage of the light bulb: ");
  scanf("%d", & watts);
  switch (watts) {
  case 15:
    lumens = 125;
    break;
  case 25:
    lumens = 215;
    break;
  case 40:
    lumens = 500;
    break;
  case 60:
    lumens = 880;
    break;
  case 75:
    lumens = 1000;
    break;
  case 100:
    lumens = 1675;
    break;
  default:
    lumens = 1;
    break;
  }
  printf("The brightness of the light bulb is %d
lumens.\n", lumens);
  return 0;
}
```

Output:

```
Enter the wattage of the
light bulb: 15
The brightness of the light
bulb is 125 lumen
```

13. Keiths Sheet Music needs a program to implement its music teachers discount policy. The program is to prompt the user to enter the purchase total and to indicate whether the purchaser is a teacher. The store plans to give each customer a printed receipt, so your program is to create a nicely formatted file called **receipt.txt**. Music teachers receive a 10% discount on their sheet music purchases unless the purchase total is $100 or higher. In that case, the discount is 12%. The discount calculation occurs before addition of the 5% sales tax. Here are two sample output filesone for a teacher and one for a nonteacher.

```
Total purchases            $122.00
Teacher's discount (12%)     14.64
Discounted total            107.36
Sales tax (5%)                5.37
Total                       $112.73
Total purchases             $24.90
Sales tax (5%)                1.25
Total                       $26.15
```

Note: to display a % sign, place two % signs in the format string:

```
printf("%d%%", SALES_TAX);
```

To write the output in the file **receipt.txt** use output redirection, **./a.out > receipt.txt**

Space for Program and Output:

Program:

```
#include <stdio.h>
#define TEACHER_DISCOUNT_RATE 0.12 // 12% discount for teachers
#define NORMAL_DISCOUNT_RATE 0.10 // 10% discount for non-teachers
#define SALES_TAX_RATE 0.05 // 5% sales tax rate
int main() {
  double total_purchases, discount, discounted_total, sales_tax,
  final_total;
  int is_teacher;
  printf("Enter the total purchases: $");
  scanf("%lf", & total_purchases);
  printf("Is the purchaser a teacher? (1 for Yes, 0 for No): ");
  scanf("%d", & is_teacher);
  if (is_teacher) {
    if (total_purchases >= 100.0) {
      discount = total_purchases * TEACHER_DISCOUNT_RATE;
    } else {
      discount = total_purchases * NORMAL_DISCOUNT_RATE;
    }
  } else {
    discount = total_purchases * NORMAL_DISCOUNT_RATE;
  }
  discounted_total = total_purchases - discount;
  sales_tax = discounted_total * SALES_TAX_RATE;
  final_total = discounted_total + sales_tax;
  printf("Total Purchase:%.2lf\n", total_purchases);
  printf("Discounted Total:%.2lf\n", discounted_total);
  printf("Sales Tax:%.2lf\n", sales_tax);
  printf("Final Total:%.2lf\n", final_total);
  return 0;
}
```

Space for Program and Output:

Output:

```
Enter the total purchases: $15000
Is the purchaser a teacher? (1 for Yes, 0 for No):
1
Total Purchase:15000.00
Discounted Total:13200.00
Sales Tax:660.00
Final Total:13860.00
```

14. A particular cell phone plan includes 50 minutes of air time and 50 text messages for $15.00 a month. Each additional minute of air time costs $0.25, while additional text messages cost $0.15 each. All cell phone bills include an additional charge of $0.44 to support 911 call centers, and the entire bill (including the 911 charge) is subject to 5 percent sales tax.

Write a program that reads the number of minutes and text messages used in a month from the user. Display the base charge, additional minutes charge (if any), additional text message charge (if any), the 911 fee, tax and total bill amount. Only display the additional minute and text message charges if the user incurred costs in these categories. Ensure that all of the charges are displayed using 2 decimal places.

Space for Program and Output:

Program:

```
#include <stdio.h>
#define BASE_CHARGE 15.00
#define ADDITIONAL_MINUTE_CHARGE 0.25
#define ADDITIONAL_TEXT_MESSAGE_CHARGE 0.15
#define NINE_ONE_ONE_FEE 0.44
#define SALES_TAX_RATE 0.05
int main() {
    int minutes_used, text_messages_used;
    double additional_minute_charge, additional_text_message_charge,
nine_one_one_fee, tax, total_bill_amount;
    printf("Enter the number of minutes used: ");
    scanf("%d", & minutes_used);
    printf("Enter the number of text messages used: ");
    scanf("%d", & text_messages_used);
    if (minutes_used > 50) {
      additional_minute_charge = (minutes_used - 50) *
ADDITIONAL_MINUTE_CHARGE;
    }
```

Space for Program and Output:

```
      else {
       additional_minute_charge = 0.0;
    }
    if (text_messages_used > 50) {
      additional_text_message_charge = (text_messages_used - 50) *
ADDITIONAL_TEXT_MESSAGE_CHARGE;
    } else {
      additional_text_message_charge = 0.0;
    }
    nine_one_one_fee = NINE_ONE_ONE_FEE;
    tax = (BASE_CHARGE + additional_minute_charge +
additional_text_message_charge + nine_one_one_fee) * SALES_TAX_RATE;
    total_bill_amount = BASE_CHARGE + additional_minute_charge +
additional_text_message_charge + nine_one_one_fee + tax;
    printf("Base charge: $%.2f\n", BASE_CHARGE);
    if (additional_minute_charge > 0.0) {
      printf("Additional minutes charge: $%.2f\n",
additional_minute_charge);
    }
    if (additional_text_message_charge > 0.0) {
      printf("Additional text message charge: $%.2f\n",
additional_text_message_charge);
    }
    printf("911 fee: $%.2f\n", nine_one_one_fee);
    printf("Tax: $%.2f\n", tax);
    printf("Total bill amount: $%.2f\n", total_bill_amount);
    return 0;
}
```

Output:

```
Enter the number of minutes used: 60
Enter the number of text messages used: 55
Base charge: $15.00
Additional minutes charge: $2.50
Additional text message charge: $0.75
911 fee: $0.44
Tax: $1.07

Total bill amount: $20.76
```

15. Write a program that determines the day number (1 to 366) in a year for a date that is provided as input data. As an example, January 1, 1994, is day 1. December 31, 1993, is day 365. December 31, 1996, is day 366, since 1996 is a leap year. A year is a leap year if it is divisible by four, except that any year divisible by 100 is a leap year only if it is divisible by 400. Your program should accept the month, day, and year as integers. Include a function leap that returns 1 if called with a leap year, 0 otherwise.

Space for Program and Output:

Program:

```
#include <stdio.h>
int leap(int year) {
  if (year % 400 == 0) {
    return 1;
  } else if (year % 100 == 0) {
    return 0;
  } else if (year % 4 == 0) {
    return 1;
  } else {
    return 0;
  }
}
int day_number(int month, int day, int year) {
  int days_in_year = 365;
  if (leap(year)) {
    days_in_year++;
  }
  int days_in_previous_months = 0;
  for (int i = 1; i < month; i++) {
    days_in_previous_months += (i == 2 && leap(year)) ? 29 : (i < 8) ?
31 : 30;
  }
  int day_number = days_in_previous_months + day;
  return day_number;
}

int main() {
  int month, day, year;
  printf("Enter the month: ");
  scanf("%d", & month);
  printf("Enter the day: ");
  scanf("%d", & day);
  printf("Enter the year: ");
  scanf("%d", & year);
  int day_number = day_number(month, day, year);
  printf("The day number is %d.\n", day_number);
  return 0;
}
```

Space for Program and Output:

Output:

```
Enter the month: 1
Enter the day: 1
Enter the year: 1994
The day number is 1.

Enter the month: 12
Enter the day: 31
Enter the year: 1993
The day number is 365.

Enter the month: 12
Enter the day: 31
Enter the year: 1996
The day number is 366.
```

16. A triangle can be classified based on the lengths of its sides as equilateral, isosceles or scalene. All three sides of an equilateral triangle have the same length. An isosceles triangle has two sides that are the same length, and a third side that is a different length. If all of the sides have different lengths then the triangle is scalene. Write a program that reads the lengths of the three sides of a triangle from theuser. Then display a message that states the triangles type.

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
  int side1, side2, side3;
  printf("Enter the lengths of the three sides of the triangle: ");
  scanf("%d %d %d", & side1, & side2, & side3);
  if (side1 == side2 && side2 == side3) {
    printf("The triangle is equilateral.\n");
  } else if (side1 == side2 || side1 == side3 || side2 == side3) {
    printf("The triangle is isosceles.\n");
  } else {
    printf("The triangle is scalene.\n");
  }
  return 0;
}
```

Space for Program and Output:

Output:

```
Enter the lengths of the three sides of the triangle: 5 5 5
The triangle is equilateral.
Enter the lengths of the three sides of the triangle: 5 5 6
The triangle is isosceles.
Enter the lengths of the three sides of the triangle: 5 6 7
The triangle is scalene.
```

Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

WEEK-END ASSIGNMENT-03

Operating Systems Workshop (CSE 3541)

Problem Statement:

Working with repetition control structure (**`for`**, **`while`** and **`do-while`**) in programming.

Assignment Objectives:

To learn how to use C repetition control structure in programming and when to use each type in developing programs.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realise the outcome of that question in relevant to systems programming. You may use additional pages on requirement.

Programming/ Output Based Questions:

1. Find the output/ error of following code snippet

Code snippet: 1(a)

```
int i =  0;
while (i <= 5) {
  printf("%3d %3d\n", i, 10 - i);
  i = i + 1;
 }
```

Output

```
0  10
1   9
2   8
3   7
4   6
5   5
```

Code snippet: 1(b)

```
int i=1;
while ( )
{
  printf ( "%d ", i++ ) ;
  if(i>10)
    break ;
}
```

Output

```
Error: expected expression
before ')' token

In the while loop we must
need a condition to execute
the program
```

Code snippet: 1(c)

```
int a=1;
do {
  printf("%d ", a++);
 } while ( a < 10 );
```

Output

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

Code snippet: 1(d)

```
int i, j,n=5;
for(i=1, j=1; j<= n; i+= 2, j++){
  printf("%d%d\n", i, j);
}
```

Output

```
11
32
53
74
95
```

Code snippet: 1(e)

```
int count = 11;
while (--count+1);
  printf("count down is %d \n",count);
```

Output

```
count down is -1
```

Code snippet: 1(f)

```
float x = 1.1 ;
while ( x == 1.1 ) {
        printf ( "%f\n", x ) ;
        x = x - 0.1 ;
}
```

Output

```
1.1
```

2. During execution of the following program segment, how many lines of hash marks are displayed?

```
for (m = 9; m > 0; --m)
  for (n = 6; n > 1; --n)
    printf("#########\n");
```

Output

```
45 lines of ‘#’ marks are
displayed
```

3. During execution of the following program segment:

   a. How many times does the first call to **printf** execute?
   b. How many times does the second call to **printf** execute?
   c. What is the last value displayed?

```
for (m = 10; m > 0; --m){
  for (n = m; n > 1; --n)
    printf("%d ", m + n);
  printf("\n");
}
```

Output

```
a. 10 times

b. 9 times

c. The last value displayed is 1
```

4. An integer n is divisible by 9 if the sum of its digits is divisible by 9. Develop a program to display each digit, starting with the rightmost digit. Your program should also determine whether or not the number is divisible by 9. Test it on the following numbers:
   n = 154368
   n = 621594
   n = 123456
   Hint: Use the % operator to get each digit; then use / to remove that digit. So 154368 % 10 gives 8 and 154368 / 10 gives 15436. The next digit extracted should be 6, then 3 and so on.

Space for Program and Output:

Program:

```
#include <stdio.h>

int main() {
  int n, sum, digit;
  printf("Enter a number: ");
  scanf("%d", & n);

  sum = 0;
  while (n > 0) {
    digit = n % 10;
    sum += digit;
    n /= 10;
  }

  if (sum % 9 == 0) {
    printf("%d is divisible by 9.\n", n);
  } else {
    printf("%d is not divisible by 9.\n", n);
  }

  printf("The digits of %d, starting with the
rightmost digit, are: ", n);
  while (n > 0) {
    digit = n % 10;
    printf("%d ", digit);
    n /= 10;
  }

  printf("\n");

  return 0;
}
```

Output:

```
Enter a number: 154368
154368 is divisible by
9.
The digits of 154368,
starting with the
rightmost digit, are: 8
6 3 4 5 1
```

5. The greatest common divisor (gcd) of two integers is the product of the integers common factors. Write a program that inputs two numbers and implements the following approach to finding their gcd. We will use the numbers ~~2~~52 and 735. Working with the numbers' absolute values, we find the remainder of one divide by the other.

```
735 |   0
    |  252
    -  0
    |  252
```

Now we calculate the remainder of the old divisor divided by the remainder found.

```
252 |   2
    |  735
    -  504
    |  231
```

We repeat this process until the remainder is zero.

```
231 |   1
    |  252
    -  231
    |   21
```

➜

```
 21 |   11
    |  231
    -  21
    |   21
    -  21
    |    0
```

The last divisor (21) is the gcd.

Space for Program and Output:

Program:

```
#include <stdio.h>
#include <stdlib.h>
int gcd(int a, int b) {
  a = abs(a);
  b = abs(b);
  while (b != 0) {
    int remainder = a % b;
    a = b;
    b = remainder;
  }
  return a;
}
```

Space for Program and Output:

Program:

```
int main() {
  int a, b;
  printf("Enter two numbers: ");
  scanf("%d %d", & a, & b);
  int gcd = gcd(a, b);
  printf("The gcd of %d and %d is %d.\n", a, b, gcd);
  return 0;
}
```

Output:

```
Enter two numbers: -252 735
The gcd of -252 and 735 is 9.
```

6. Write a program to process a collection of the speeds of vehicles. Your program should count and print the number of vehicles moving at a high speed (90 miles/hour or higher), the number of vehicles moving at a medium speed (50-89 miles/hour), and the number of vehicles moving at a slow speed (less than 50 miles/hour). It should also display the category of each vehicle. Test your program on the following data in a file:
43 23 54 57 68 67 51 90 33 22 11 88 34 52 75 12 78 32 89 14 65 67 97 53 10 47 34

   a. Also code to display the average speed of a category of vehicle (a real number) at the end of the run.

   b. Store the data into a file **`vspeed.txt`**. *Use input redirection to read all numbers from that file. (i.e. **`./a.out < vspeed.txt`**)*

   c. While reading the input from the file, apply the idea of **`scanf`** function returns. *The **`scanf`** returns: (1) On success, this function returns the number of input items successfully matched and assigned (ii) The value **EOF** is returned if the end of input is reached before either the first successful conversion or a matching failure occurs.*

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
    int high = 0, medium = 0, low = 0;
    int high_total = 0, medium_total = 0, low_total = 0;
    int count = 0;
    int speed;

    while (scanf("%d", &speed) != EOF) {
        if (speed >= 90) {
            high++;
            high_total += speed;
        } else if (speed >= 50) {
            medium++;
            medium_total += speed;
        } else {
            low++;
            low_total += speed;
        }
        printf("Speed: %d belongs to ",speed);
        printf("Category: ");
        if (speed >= 90) {
            printf("High Speed. \n");
        } else if (speed >= 50) {
            printf("Medium Speed. \n");
        } else {
            printf("Low Speed. \n");
        }
        count++;
    }
```

Space for Program and Output:

Program:

```
    printf("Number of vehicles at high speed: %d\n", high);
    printf("Number of vehicles at medium speed: %d\n", medium);
    printf("Number of vehicles at low speed: %d\n", low);
    printf("Average high-speed: %.2f mph\n", (high_total * 1.0) / high);
    printf("Average medium-speed: %.2f mph\n", (medium_total * 1.0) /
medium);
    printf("Average low-speed: %.2f mph\n", (low_total * 1.0) / low);
    return 0;
}
```

Output:

```
Speed: 43 belongs to Category: Low Speed.
Speed: 23 belongs to Category: Low Speed.
Speed: 54 belongs to Category: Medium Speed.
Speed: 57 belongs to Category: Medium Speed.
Speed: 68 belongs to Category: Medium Speed.
Speed: 67 belongs to Category: Medium Speed.
Speed: 51 belongs to Category: Medium Speed.
Speed: 90 belongs to Category: High Speed.
Speed: 33 belongs to Category: Low Speed.
Speed: 22 belongs to Category: Low Speed.
Speed: 11 belongs to Category: Low Speed.
Speed: 88 belongs to Category: Medium Speed.
Speed: 34 belongs to Category: Low Speed.
Speed: 52 belongs to Category: Medium Speed.
Speed: 75 belongs to Category: Medium Speed.
Speed: 12 belongs to Category: Low Speed.
Speed: 78 belongs to Category: Medium Speed.
Speed: 32 belongs to Category: Low Speed.
Speed: 89 belongs to Category: Medium Speed.
Speed: 14 belongs to Category: Low Speed.
Speed: 65 belongs to Category: Medium Speed.
Speed: 67 belongs to Category: Medium Speed.
Speed: 97 belongs to Category: High Speed.
Speed: 53 belongs to Category: Medium Speed.
Speed: 10 belongs to Category: Low Speed.
Speed: 47 belongs to Category: Low Speed.
Speed: 34 belongs to Category: Low Speed.
Number of vehicles at high speed: 2
Number of vehicles at medium speed: 13
Number of vehicles at low speed: 12
Average high-speed: 93.50 mph
Average medium-speed: 66.46 mph
Average low-speed: 26.25 mph
```

7. A baseball player's batting average is calculated as the number of hits divided by the official number of at-bats. In calculating official at-bats, walks, sacrifices, and occasions when hit by the pitch are not counted. Write a program that takes an input file containing player numbers and batting records. Trips to the plate are coded in the batting record as follows: H-hit, O-out, W-walk, S-sacrifice, P-hit by pitch. The program should output for each player the input data followed by the batting average. Each batting record is followed by a newline character. **Your program should not use any kind of array and array processing**.

Sample input file:

```
12 HOOOWSHHOOHPWWHO
4 OSOHHHWWOHOHOOO
7 WPOHOOHWOHHOWOO
```

Corresponding output:

```
Player 12's record: HOOOWSHHOOHPWWHO
Player 12's batting average: 0.455
Player 4's record: OSOHHHWWOHOHOOO
Player 4's batting average: 0.417
Player 7's record: WPOHOOHWOHHOWOO
Player 7's batting average: 0.364
```

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
    FILE *file = fopen("Q7input.txt", "r");
    if (file == NULL) {
        printf("Error opening the file.\n");
        return 1;
    }
    int player;
    char record;
    double average;
    while (fscanf(file, "%d ", &player) == 1) {
        printf("Player %d's record: ", player);
        int hits = 0;
        int at_bats = 0;
        while (1) {
            if (fscanf(file, "%c", &record) != 1 || record == '\n') {
                break;
            }
            if (record == 'H') {
                hits++;
                at_bats++;
                putchar('H');
            } else if (record == 'O') {
                at_bats++;
                putchar(record);
            } else {
                putchar(record);
            }
        }
```

Space for Program and Output:

Program:

```
          if (at_bats > 0) {
            average = (double)hits / at_bats;
        } else {
            average = 0.0;
        }
      printf("\nPlayer %d's batting average: %.3f\n", player, average);
    }
    fclose(file);
    return 0;
}
```

Output:

```
Player 12's record: HOOOHHOOHHO
Player 12's batting average: 0.455
Player 4's record: OOHHHOHOHOOO
Player 4's batting average: 0.417
Player 7's record: OHOOHOHHOOO
Player 7's batting average: 0.364
```

8. Write a program to process a collection of scores obtained by students of a class of certain strength. Your program should count and print the number of students with Grade A (80 and higher), Grade B(65-79), Grade C(40-64) and Grade F(39 and below). Ensure that the entered scores must remain in between 0 and 100(inclusive). Test your program on the following data:

```
8
23 67 65 12
89 32 17 45
41 58 60 78
82 88 19 22
70 88 41 89
78 79 72 68
74 59 75 81
44 59 23 12
```

   a. Read the same input from a file using input redirection. First line represents number of students and rest of the lines represent the marks obtained by each student in 4 subjects.

   b. Display average score and grade of each student in form of a table.(***Hint:*** *Average score of a student=*$(m_1 + m_2 + m_3 + m_4)/4$*. where* m_i*,* $i = 1, 2, 3, 4$ *represent mark in subject i and calculate grade according to the specified condition given above.*

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
    int numStudents, score1, score2, score3, score4;
    int countA = 0, countB = 0, countC = 0, countF = 0;
    scanf("%d", &numStudents);
    printf("Student\tAverage\tGrade\n");
    for (int i = 1; i <= numStudents; i++) {
        int average;
        char grade;
        scanf("%d %d %d %d", &score1, &score2, &score3, &score4);
        average = (score1 + score2 + score3 + score4) / 4;
        if (average >= 80) {
            grade = 'A';
            countA++;
        } else if (average >= 65) {
            grade = 'B';
            countB++;
        } else if (average >= 40) {
            grade = 'C';
            countC++;
        } else {
            grade = 'F';
            countF++;
        }
        printf("%d\t%d\t%c\n", i, average, grade);
    }
```

Space for Program and Output:

Program:

```
    printf("Grade A: %d students\n", countA);
    printf("Grade B: %d students\n", countB);
    printf("Grade C: %d students\n", countC);
    printf("Grade F: %d students\n", countF);

    return 0;
}
```

Output:

```
Student    Average    Grade
1          41         C
2          45         C
3          59         C
4          52         C
5          72         B
6          74         B
7          72         B
8          34         F

Grade A: 0 students
Grade B: 3 students
Grade C: 4 students
Grade F: 1 students
```

9. Design a C program to display the following pattern;

```
A B C D E F G F E D C B A
A B C D E F   F E D C B A
A B C D E       E D C B A
A B C D           D C B A
A B C               C B A
A B                   B A
A                       A
```

Space for Program and Output:

Program:

```
#include <stdio.h>
int main() {
    int n = 7;
    for (int i = 0; i < n; i++) {
        for (char ch = 'A'; ch < 'A' + n - i; ch++) {
            printf("%c ", ch);
        }
        for (int space = 0; space < 2 * i; space++) {
            printf("  ");
        }
        for (char ch = 'A' + n - i - 1; ch >= 'A'; ch--) {
            if (ch != 'A') {
                printf("%c ", ch);
            } else {
                printf("%c\n", ch);
            }
        }
    }

    return 0;
}
```

Output:

```
A B C D E F G G F E D C B A
A B C D E F     F E D C B A
A B C D E         E D C B A
A B C D             D C B A
A B C                 C B A
A B                     B A
A                         A
```

10. The natural logarithm can be approximated by the following series.

$$\frac{x-1}{x}+\frac{1}{2}\left(\frac{x-1}{x}\right)^2+\frac{1}{2}\left(\frac{x-1}{x}\right)^3+\frac{1}{2}\left(\frac{x-1}{x}\right)^4+\ldots$$

If x is input through the keyboard, write a program to calculate the sum of first nine terms of this series.

Space for Program and Output:

Program:

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

int main() {
    float x;
    printf("Enter a value for x: ");
    scanf("%f", & x);
    float sum = 0.0 f;
    for (int i = 1; i <= 9; i++) {
        sum += (x - 1.0 f / x) * powf(x - 1.0 f / x, i) / i;
    }
    printf("The sum of the first nine terms of the series is: %.4f\n", sum);
    return 0;
}
```

Output:

```
Enter a value for x: 2
The sum of the first nine terms of the series is: 27.5231
```

11. Write a menu driven program which has following options:

    1. Factorial of a number.
    2. Prime or not
    3. Odd or even
    4. Exit

Use input-validation loop and program should terminate only when option 4 is selected.

Space for Program and Output:

Program:

```
#include <stdio.h>
int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
int isPrime(int n) {
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}
int isEven(int n) {
    return (n % 2 == 0);
}

int main() {
    int option;
    int number;
    do {
        printf("Menu:\n");
        printf("1. Factorial of a number\n");
        printf("2. Prime or not\n");
        printf("3. Odd or even\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", & option);
        if (option < 1 || option > 4) {
            printf("Invalid option. Please enter again.\n");
        } else {
            switch (option) {
                case 1:
                    printf("Enter a number: ");
                    scanf("%d", & number);
```

Space for Program and Output:

Program:

```
                    printf("The factorial of %d is %d.\n", number,
factorial(number));
                    break;
                case 2:
                    printf("Enter a number: ");
                    scanf("%d", & number);
                    if (isPrime(number)) {
                        printf("%d is a prime number.\n", number);
                    } else {
                        printf("%d is not a prime number.\n", number);
                    }
                    break;
                case 3:
                    printf("Enter a number: ");
                    scanf("%d", & number);
                    if (isEven(number)) {
                        printf("%d is an even number.\n", number);
                    } else {
                        printf("%d is an odd number.\n", number);
                    }
                    break;
                case 4:
                    printf("Exiting the program.\n");
                    break;
                }
            }
        } while (option != 4);
    return 0;
}
```

Output:

```
Menu:
1. Factorial of a number
2. Prime or not
3. Odd or even
4. Exit
Enter your choice: 1
Enter a number: 5
The factorial of 5 is 120.
Menu:
1. Factorial of a number
2. Prime or not
3. Odd or even
4. Exit
Enter your choice: 2
Enter a number: 13
13 is a prime number.
```

Space for Program and Output:

Output:

```
Menu:
1. Factorial of a number
2. Prime or not
3. Odd or even
4. Exit
Enter your choice: 3
Enter a number: 28
28 is an even number.
Menu:
1. Factorial of a number
2. Prime or not
3. Odd or even
4. Exit
Enter your choice: 4
Exiting the program.
```

WEEK-END ASSIGNMENT-04

Top-Down Design with Functions

Operating Systems Workshop (CSE 3541)

Problem Statement:

Demonstrate the top-down design method of problem solving and emphasize the role of modular programming using functions.

Assignment Objectives:

To learn about functions and how to use them to write programs with separate modules.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realise the outcome of that question in relevant to systems programming. You may use additional pages on requirement.

Programming/ Output Based Questions:

1. Describe the problem input(s), output(s) and write the algorithm for a program that computes the circle's area and circumference. Also write a function prototype to compute the same using radius as input to the function and return type **`void`**.

Input, Output, Algorithm and Function Prototype:

Program and Algorithm:

```
#include <stdio.h>
#define PI 3.14159

void circle(float radius);                          //Function Prototype
int main() {
    printf("Enter Radius: ");                                //Input
    float r;
    scanf("%f",&r);
    circle(r);
    return 0;
}
void circle(float radius){
    float area = PI*radius*radius;
    float circum = 2*PI*radius;
    printf("Area of a circle: %.2f\n",area);               //Output
    printf("Circumference of a circle: %.2f\n",circum);    //Output

}
```

Output:

```
Enter Radius: 5
Area of a circle: 78.54
Circumference of a circle: 31.42
```

2. During execution of the following program segment, how many lines of star marks will be displayed?

```
void nonsense(void){
  printf("*****\n");
  printf("* *\n");
  printf("*****\n");
}

int main(void){
 nonsense();
 nonsense();
 nonsense();
 return (0);
}
```

Output:

```
*****
* *
*****
*****
* *
*****
*****
* *
*****
```

3. Consider the following C program : **[GATE 2004]**

```
int main(void){
 int x,y,m,n;
 scanf("%d%d",&x,&y);
 /* Assume x > 0 and y > 0  */
 m = x; n = y;
 while(m!=n){
   if(m>n)
       m=m-n;
   else
       n=n-m;
 }
 printf("%d",n);
}
```

The program computes :

(a) x + y using repeated subtraction
(b) x mod y using repeated subtraction
(c) the greatest common divisor of x and y
(d) the least common multiple of x and y

Output/Answer:

(c) the greatest common divisor of x and y

4. What does the following algorithm approximate? (Assume $m > 1$, $e > 0$). **[GATE 2004]**

```
x = m; y = 1;
while (x - y > e){
  x = (x + y) / 2;
  y = m/x;
}
printf("%d",x);
```

The program computes

(a) $log\ m$ (c) $m^{\frac{1}{2}}$

(b) m^2 (d) $m^{\frac{1}{3}}$

Output/Answer:

(c) $m^{1/2}$

5. Consider the following two functions. **[GATE 2017]**

Find the output when **`fun1(15)`** is called;

```
void fun1(int n)
{
  if(n==0)
     return;
  printf("%d",n);
  fun2(n-2);
  printf("%d",n);
}
```

```
void fun2(int n)
{
  if(n==0)
     return;
  printf("%d",n);
  fun1(++n);
  printf("%d",n);
}
```

Output/Answer:

(a) 53423122222445

6. The output of executing the following C program is; **GATE**

```
int total(int v){
 int count=0;
 while(v){
  count +=v&1;
  v>>=1;
 }
 return count;
}
```

```
int main(){
int x=0,i=5;
for( ; i>0; i--){
   x=x+total(i);
}
printf("%d\n",x);
return 0;
}
```

Output:
7

How many times the function call, **total( )**, is called?

Ans: 5 times.

7. Consider the following C function;

```
int
 fun(n){
 int i,j;
 for(i=1;i<=n;i++){
   for(j=1;j<n;j++){
      printf("%d %d\n",i,j);
   }
 }
 return 1;
}
```

Output:

Determine the number of times the printf() will be executed if n=5.

Ans: 16 times.

8. Write a program that prompts the user for the two legs of a right triangle and makes use of the **`pow`** and **`sqrt`** functions and the Pythagorean theorem to compute the length of the hypotenuse.

Space for Program and Output:

Program:

```
#include <stdio.h>
#include <math.h>
double hypotenuse(double x, double y) {
    double z = sqrt(pow(x, 2) + pow(y, 2));
    return z;
}
int main(void) {
    double b1, b2;
    printf("Enter side 1: ");
    scanf("%lf", &b1);
    printf("Enter side 2: ");
    scanf("%lf", &b2);
    printf("Hypotenuse: %.2lf\n",
hypotenuse(b1, b2));
    return 0;
}
```

Output:

```
Enter side 1: 3
Enter side 2: 4
Hypotenuse: 5.00
```

9. Write the prototype for a function called **`script`** that has three input parameters. The first parameter will be the number of spaces to display at the beginning of a line. The second parameter will be the character to display after the spaces, and the third parameter will be the number of times to display the second parameter on the same line and return type of the function is of integer.

Function Prototype/ Declaration/ Signature:

```
#include <stdio.h>
int script(int num_spaces, char char_to_display, int num_times);
```

10. In a particular jurisdiction, taxi fares consist of a base fare of $4.00, plus $0.25 for every 140 meters traveled. Write a function that takes the distance traveled (in kilometers) as its only parameter and returns the total fare as its only result. Write a main program that demonstrates the function.
Hint: Taxi fares change over time. Use constants to represent the base fare and the variable portion of the fare so that the program can be updated easily when the rates increase.

Space for Program and Output:

Program:

```
#include <stdio.h>

#define BASE_FARE 4.00
#define RATE_PER_KILOMETER 0.25
#define METERS_PER_KILOMETER 1000
#define METERS_PER_FARE_UNIT 140

double calculateTaxiFare(double
distanceInKilometers) {
      double distanceInMeters =
distanceInKilometers * METERS_PER_KILOMETER;
      double fare = BASE_FARE +
      (distanceInMeters / METERS_PER_FARE_UNIT)
* RATE_PER_KILOMETER;
      return fare;
}
int main() {
      double distanceInKilometers;
      printf("Enter the distance traveled (in
kilometers): ");
      scanf("%lf", &distanceInKilometers);
      double fare =
calculateTaxiFare(distanceInKilometers);
      printf("Total Fare: $%.2lf\n", fare);
      return 0;
}
```

Output:

```
Enter the distance
travelled (in kilometers):
25
Total Fare: $48.64
```

11. Create a function named **nextPrime** that finds and returns the first prime number larger than some integer, **n**. The value of **n** will be passed to the function as its only parameter. The main function in the program that reads an integer from the user and displays the first prime number larger than the entered value. Additionally, your program must specify the function prototype and identify the actual argument(s) and formal parameters.

Space for Program and Output:

Program:

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

int nextPrime(int n);

bool isPrime(int num) {
      if (num <= 1)
            return false;
      for (int i = 2; i * i <= num; i++) {
            if (num % i == 0)
                  return false;
      }
      return true;
}

int main() {
      int n;
      printf("Enter an integer: ");
      scanf("%d", &n);
      int next = nextPrime(n);
      printf("The first prime number larger
than %d is: %d\n", n, next);
      return 0;
}

int nextPrime(int n) {
      n++;
      while (true){
            if (isPrime(n))
                  return n;
      n++;
      }
}
```

Output:

```
Enter an integer: 26
The first prime number
larger than 26 is: 29
```

12. Write a program that allows the user to convert a number from one base to another. Your program should support bases between 2 and 16 for both the input number and the result number. If the user chooses a base outside of this range then an appropriate error message should be displayed and the program should exit. Divide your program into several functions, including a function that converts from an arbitrary base to base 10, a function that converts from base 10 to an arbitrary base, and a main program that reads the bases and input number from the user.

Space for Program and Output:

Program:

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

int fromArbitraryBaseToDecimal(const char * number, int base) {
    int decimal = 0;
    int length = strlen(number);
    int power = 1;
    for (int i = length - 1; i >= 0; i--) {
        int digit = (isdigit(number[i]) ? (number[i] - '0') : (number[i] - 'A'
+ 10));
        if (digit >= base) {
            printf("Error: Invalid digit for the base.\n");
            return -1;
        }
        decimal += digit * power;
        power *= base;
    }
    return decimal;
}

void fromDecimalToArbitraryBase(int decimal, int base) {
    char result[32];
    int index = 0;
    while (decimal > 0) {
        int remainder = decimal % base;
        result[index++] = (remainder < 10) ? (remainder + '0') : (remainder -
10 + 'A');
        decimal /= base;
    }
    if (index == 0) {
        printf("0\n");
        return;
    }
    printf("Converted number: ");
    for (int i = index - 1; i >= 0; i--) {
        printf("%c", result[i]);
    }
    printf("\n");
}
```

Space for Program and Output:

Program:

```
    int main() {
    int inputBase, outputBase;
    char number[32];
    printf("Enter the input base (2-16): ");
    scanf("%d", & inputBase);
    printf("Enter the output base (2-16): ");
    scanf("%d", & outputBase);
    if (inputBase < 2 || inputBase > 16 || outputBase < 2 || outputBase
   > 16) {
        printf("Error: Bases must be between 2 and 16.\n");
        return 1; // Exit with an error
    }
    printf("Enter the number in base-%d: ", inputBase);
    scanf("%s", number);
    int decimal = fromArbitraryBaseToDecimal(number, inputBase);
    if (decimal != -1) {
        fromDecimalToArbitraryBase(decimal, outputBase);
    }
    return 0;
}
```

Output:

```
Enter the input base (2-16): 2
Enter the output base (2-16): 8
Enter the number in base-2: 0100110
Converted number: 46

Enter the input base (2-16): 16
Enter the output base (2-16): 8
Enter the number in base-16: AB158D
Converted number: 52612615
```

13. Write a program that calculates the speed of sound (a) in air of a given temperature T ($^{\circ}F$). Formula to compute the speed in ft/sec:

$$a = 1086\sqrt{\frac{5T + 297}{247}}$$

Be sure your program does not lose the fractional part of the quotient in the formula shown. As part of your solution, write and call a function that displays instructions to the program user.

Space for Program and Output:

Program:

```
#include <stdio.h>
#include <math.h>
void displayInstructions() {
    printf("This program calculates the speed of sound in air based on the
given temperature in degrees Fahrenheit.\n");
    printf("Please enter the temperature in degrees Fahrenheit when
prompted.\n");
    printf("The result will be displayed in feet per second(ft/sec).\n");
}
double calculateSpeedOfSound(double temperature) {
    return 1086 * sqrt((5 * temperature + 297) / 247);
}
int main() {
    double temperature;
    displayInstructions();
    printf("Enter the temperature in degrees Fahrenheit: ");
    scanf("%lf", &temperature);
    double speedOfSound = calculateSpeedOfSound(temperature);
    printf("The speed of sound in air at %.2lf°F is %.2lf ft/sec.\n",
temperature, speedOfSound);
    return 0;
}
```

Output:

```
This program calculates the speed of sound in air based on the given
temperature in degrees Fahrenheit.
Please enter the temperature in degrees Fahrenheit when prompted.
The result will be displayed in feet per second(ft/sec).
Enter the temperature in degrees Fahrenheit: 65
The speed of sound in air at 65.00°F is 1723.36 ft/sec.
```

14. After studying the population growth of Gotham City in the last decade of the 20th century, we have modeled Gotham's population function as

$$P(t) = 52.966 + 2.184t$$

where t is years after 1990, and P is population in thousands. Thus, $P(0)$ represents the population in 1990, which was 52.966 thousand people. Write a program that defines a function named population that predicts Gotham's population in the year provided as an input argument. Write a program that calls the function and interacts with the user as follows:

```
Enter a year after 1990> 2015
Predicted Gotham City population for 2010 (in thousands): 107.566
```

Space for Program and Output:

Program:

```
#include <stdio.h>
double population(int year) {
    return 52.966 + 2.184 * (year - 1990);
}
int main() {
    int year;
    printf("Enter a year after 1990: ");
    scanf("%d", &year);
    if (year < 1990) {
    printf("Please enter a year after 1990.\n");
    } else {
    double predictedPopulation = population(year);
    printf("Predicted Gotham City population for %d (in thousands):%.3lf\n",
year, predictedPopulation);
    }
    return 0;
}
```

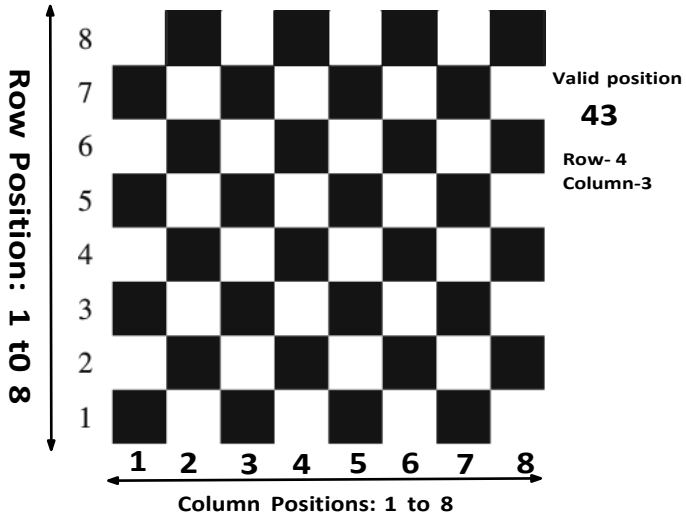
Output:

```
Enter a year after 1990: 2003
Predicted Gotham City population for 2003 (in thousands):81.358

Enter a year after 1990: 2023
Predicted Gotham City population for 2023 (in thousands):125.038

Enter a year after 1990: 2060
Predicted Gotham City population for 2060 (in thousands):205.846
```

15. Positions on a chess board are identified by a two-digit number from 11 to 88. The unit place digitidentifies the column, while the **10th** place digit identifies the row, as shown below:



Write a program that reads a position (i.e. 2 digit number) from the user. Write a use-defined function to check whether 2-digit position is a valid cell or not as per the function prototype **`int flag=IsvalidPosition (int);`** If the position is valid, use an if statement to determine if the column begins with a black square or a white square then report the color of the square in that row. For example, if the user enters **11** then your program should report that the square is black. If the user enters **34** then your program should report that the square is white.

Space for Program and Output:

Program:

```
#include <stdio.h>
int IsValidPosition(int position) {
    int row = position / 10;
    int column = position % 10;
    return (row >= 1 && row <= 8) && (column >= 1 && column <= 8);
}
int main() {
    int position;
    printf("Enter a two-digit position (11 to 88): ");
    scanf("%d", &position);
    if (IsValidPosition(position)) {
        int column = position % 10;
        if ((position / 10 + column) % 2 == 0) {
            printf("The square is black.\n");
        } else {
        printf("The square is white.\n");
        }
     }
```

Space for Program and Output:

Program:

```
     else {
        printf("Invalid position. Please enter a valid position (11 to
88).\n");
    }
    return 0;
}
```

Output:

```
Enter a two-digit position (11 to 88): 35
The square is black.

Enter a two-digit position (11 to 88): 47
The square is white.

Enter a two-digit position (11 to 88): 89
Invalid position. Please enter a valid position (11 to 88).

Enter a two-digit position (11 to 88): 16
The square is white.
```