SIKSHA 'O' ANUSANDHAN

DEEMED TO BE UNIVERSITY

Admission Batch : 2021 **Session: Odd-2023**

Laboratory Record

Operating Systems Workshop (CSE 3541)

Submitted by

Name : Amit Aryan

Registration No.: 2141001065

Branch : Computer Science & Engineering

Semester: 5th **Section:** N

Department of Computer Science & Engineering

Faculty of Engineering & Technology (ITER)

Jagamohan Nagar, Jagamara, Bhubaneswar, Odisha - 751030

Contents

LABORATORY CONFIGURATION

- **Laboratory name:** Computer Networking Workshop (CSE 4541)
- **Laboratory room No:** D-101
- **Total number of computers:** 70
- **Laboratory Machine Specifiction:**
  - ✷
  - ✷
  - ✷
  - ✷
- **Operating system:**
- **NIC info (`$lshw`):**
  - ✷
  - ✷
  - ✷
  - ✷
- **Simulator info:**
  - ✷
  - ✷
  - ✷
  - ✷
- **Availability of credentials to access server:** YES
- **Any other (if any):**

WEEK-END ASSIGNMENT-01

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

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

Write/paste output

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

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

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

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

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

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

Write/paste output in exact form as expected

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

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.

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

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

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

State the output in exact form

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

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

State the output in exact form

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

```
int main()
{
```

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

State the output in exact form

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

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

State the output in exact form

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;
}
```

State the output in exact form

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;
}
```

State the output in exact form

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

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

State the output in exact form

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

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

```
 return 0;
}
```

State the output in exact form

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

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

State the output in exact form

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

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

State the output in exact form

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

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

State the output in exact form

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

```
int main()
```

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

State the output in exact form

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

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

State the output in exact form

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

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:

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:

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:

26. Predict the output of the given code snippet;

```
int main()
{
    int num1=0,num2=0,num3=0;
    printf("Enter a number:");
```

```
    scanf("%2d%3d%4d",&num1,&num2,&num3);
    printf("%d %d %d",num1,num2,num3);
    return 0;
}
```

State the output in exact form

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

(A) 2345 9 76 456

(B) 23 9 76 45

(C) 456 76 9 2345

(D) No output

Answer with 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:

28. Choose the output of the code snippet;

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

State the output in exact form

(A) 10 10 6

(C) 6 6 6

(B) 10 10 10

(D) No output

Answer with reason:

29. A C program contains the following form; Suppose that the following string has been assigned to text `Programming with C cab be a challenging creative activity`. Show the output resulting from the following printf statements

```
int main()
{
    chat 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

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)

State the output in exact form

Code for (ii)

State the output in exact form

Code for (iii)

31. Describe the output of the code snippet;

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

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

WEEK-END ASSIGNMENT-02

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 ▼

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_
```

Answer here ▼				

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 ▼**

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 ▼

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 ▼

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)	
(b)	

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

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)
(b)
(c)
(d)

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

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)

(b)

(c)

(d)

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

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 of the 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

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

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

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

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

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

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

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

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 30 m1=

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

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

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 =

(2) B =

(3) x =

(4) p=    q=    r=
```

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

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

26. State the output of the following code snippet;

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

Answer

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

28. State the output of the following code snippet;

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

Answer

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

30. Find the output of the following code snippet;

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

Answer

31. Find the output of the following code snippet;

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

Answer

WEEK-END ASSIGNMENT-03

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 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 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

Code snippet: 1(b)

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

Output

Code snippet: 1(c)

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

Output

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

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

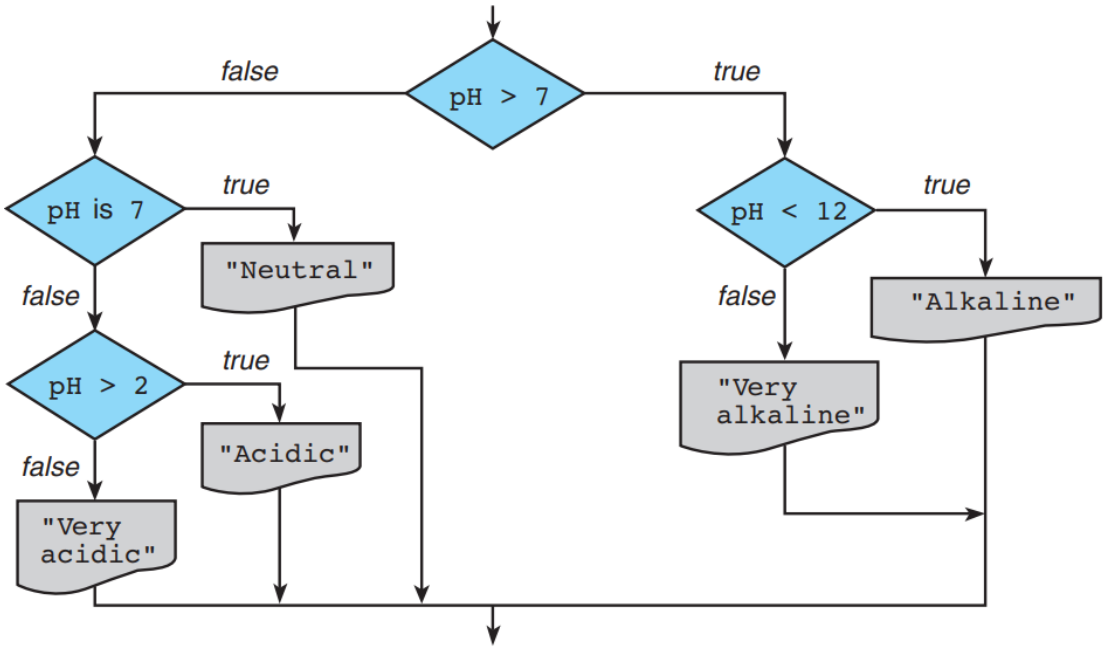
```
if (deduct < balance);
    balance = balance - deduct;
    printf("New balance is %.2f\n", balance);
 else;
    printf("Deduction of %.2f refused.\n",
        deduct);
    printf("Would overdraw account.\n");

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

Output

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 ▼

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 ▼

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 semesterregistration 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 ▼

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 ▼

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

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

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

Test cases and output ▼

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

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 in if statement

Test cases and output ▼

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

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 in the 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 ▼

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 ▼

Space for Program and output ▼

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 ▼

Space for Program and output ▼

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 ▼

Space for Program and output ▼

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 the user. Then display a message that states the triangles type.

Space for Program and output ▼

WEEK-END ASSIGNMENT-04

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

Code snippet: 1(b)

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

Output

Code snippet: 1(c)

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

Output

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

Code snippet: 1(e)

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

Output

Code snippet: 1(e)

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

Output

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

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.

b.

c.

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 ▼

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 -252 and 735. Working with the numbers' absolute values, we find the remainder of one divide by the other.

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

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

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

We repeat this process until the remainder is zero.

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

➜

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

The last divisor (21) is the gcd.

Space for Program and output ▼

Space for Program and output ▼

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 ▼

Space for Program and output ▼

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 ▼

Space for Program and output ▼

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 ▼

Space for Program and output ▼

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
```

Write/paste your code here ▼

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 ▼

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 ▼

Space for Program and output ▼

WEEK-END ASSIGNMENT-05

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 ▼

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▼

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) $log\ m$

(b) m^2

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

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

Output▼

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▼

(a) 53423122222445

(b) 53423120112233

(c) 53423122132435

(d) 53423120213243

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▼

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

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**.

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 ▼

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 ▼

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 **Output ▼**

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 **Output ▼**

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 **Output ▼**

Space for Program **Output ▼**

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 **Output ▼**

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

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

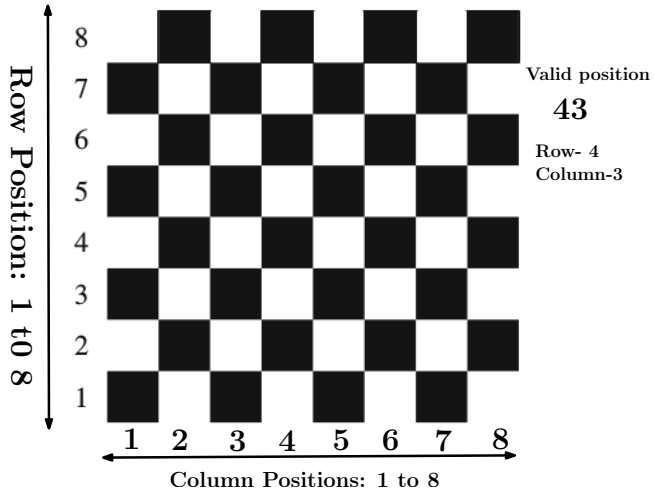
where t is years after 1990, and P is population in thousands. Thus, $P(0)$ represents the population in 1990, which was 52.966 thousand people. Write a program that defines a function named population that predicts Gotham's population in the year provided as an input argument. Write a program that calls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

Space for Program **Output ▼**

15. Positions on a chess board are identified by a two digit number from 11 to 88. The unit place digit identifies the column, while the $\mathbf{10^{th}}$ 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 protoype **`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 **Output ▼**

Space for Program **Output ▼**

WEEK-END ASSIGNMENT-06

Experiment with arrays for storing and processing collections of values of the same types in different applications.

Assignment Objectives:

To learn how to declare and use arrays for storing collections of values of the same type as well as to learn how to process the elements of an array.

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. We initialize a 25-element array with the prime numbers less than 100.

```
int prime_lt_100[] = {2, 3, 5, 7, 11, 13, 17,
    19, 23, 29, 31, 37,41, 43, 47, 53, 59,
   61, 67, 71, 73, 79, 83, 89, 97};
```

Determine the array elements given in the following expressions;

```
(a) prime_lt_100[24];
(b) int i=10; prime_lt_100[i+4];
(c) prime_lt_100[prime_lt_100[2] +
                          prime_lt_100[0]];
(d) prime_lt_100[6]=prime_lt_100[6] +
                          prime_lt_100[16];
```

Output

2. Draw the required array with it's content as per the given code snippet.

```
#define SIZE 6
int sqaure[SIZE],i;
for(i=0;i<SIZE;i++){
      square[i]=i*i;
}
```

Output

3. Consider the following C program; **[GATE 2019]**

```
int main(){
  int arr[]={1,2,3,4,5,6,7,8,9,0,1,2,5};
  int *ip=arr+4;
  printf("%d\n",ip[1]);
  return 0;
}
```

Output▼

4. State the output of the code snippet;

```
int sub();
int add(int);
int main(){
  int i;
  int arr[]={10,20,add(30),sub()+10};
  for(i=0;i<4;i++)
     printf("arr[%d]=%d\n",i,arr[i]);
  return 0;
}
int add(int x){
  return x;
}
```

```
int sub(){
  int i=10;
  return(i+30);
}
```

Output▼

5. Write the output of the code snippet;

```
int main(){
  int arr[]={1,2,3,4,5,6,7,8,9,0};
  for(int i=0;i<10;i++)
      printf("%d ",arr[i]);
  printf("\n");
  for(int i=0;i<10;i++)
      printf("%d ",i[arr]);
  printf("\n");
  for(int i=0;i<10;i++)
      printf("%d ",*(arr+i));
  printf("\n");
  for(int i=0;i<10;i++)
      printf("%d ",*(&arr[i]));
  return 0;
}
```

Output▼

6. Consider the following C program **[GATE 2019]**

```
int main(){
  int a[]={2,4,6,8,10};
  int sum=0,i,*b=a+4;
  for(i=0;i<5;i++){
     sum=sum+(*b - i) - *(b - i);
  }
  printf("Sum=%d\n",sum);
  return 0;
}
```

The output of the given program is

Output▼

7. Consider the following C function; **[GATE 2021]**

```
int SimpleFunction(int y[], int n, int x){
 int total=y[0], loopIndex;
 for(loopIndex=1;loopIndex<=n-1;loopIndex++)
    total=x*total+y[loopIndex];
 return total;
}
```

Output▼

Let $\mathbf{z}$ be an array of 10 elements with $\mathbf{z[i] = 1}$ for all $\mathbf{i}$ such that $\mathbf{0 \leq i \leq 9}$. State the value returned by the call **`SimpleFunction(z,10,2);`**

8. Write a function that takes two type int array input arguments and their effective size and produces a result array containing the sums of corresponding elements. For example, for the three-element input arrays 5 -1 7 and 2 4 -2 , the result would be an array containing 7 3 5 .

Space for Program ▼ **Output ▼**

9. The **bubble sort** is another technique for sorting an array. A bubble sort compares adjacent array elements and exchanges their values if they are out of order. In this way, the smaller values ”bubble“ to the top of the array (toward element 0), while the larger values sink to the bottom of the array. After the first pass of a bubble sort, the last array element is in the correct position; after the second pass the last two elements are correct, and so on. Thus, after each pass, the unsorted portion of the array contains one less element. Write and test a function that implements this sorting method.

Space for Program ▼ **Output ▼**

10. You have two independent sorted arrays of size m, and n respectively, where $m, n > 0$. You are required to merge the two arrays such that the merged array will be in sorted form and will contain exactly $m + n$ number of elements. You are not allowed to use any kind of sorting algorithm. Design your program to meet the above given requirement.

Example 1 :

First array:

12	20	24	32

Second array:

7	8	65	105

The merged sorted array:

7	8	12	20	24	32	65	105

Example 2 :

First array:

12	20	24

Second array:

7	8	65	105

The merged sorted array:

7	8	12	20	24	65	105

Example 3 :

First array :

12	20	24	100	120	130

Second array :

17	28	105	110

The merged sorted array:

12	17	20	24	100	105	110	120	130

NOTE :

Assume the elements of the array are non-negative integers. The elements can be read from the keyboard or can be generated randomly.

Space for Program **Output ▼**

Space for Program **Output ▼**

11. The *binary search* algorithm that follows may be used to search an array when the elements are in order. The algorithm for binary search given as;

    1. Let **bottom** be the subscript of the initial array element.
    2. Let **top** be the subscript of the last array element.
    3. Let **found** be false.
    4. Repeat as long as **bottom** isn't greater than **top** and the target has not been found
        5. Let **middle** be the subscript of the element halfway between **bottom** and **top**.
        6. if the element at **middle** is the target
                7. Set **found** to true and **index** to **middle**.

        else if the element at **middle** is larger than the target
                8. Let **top** be **middle - 1**.

        else
                9. Let **bottom** be **middle + 1**.

    Write and test a function **binary_srch** that implements this algorithm for an array of integers. When there is a large number of array elements, which function do you think is faster: **binary_srch** or the linear search algorithm.

Space for Program **Output ▼**

Space for Program **Output ▼**

12. Implement the following algorithm for linear search that sets a flag (for loop control) when the element being tested matches the target.

    1. Assume the target has not been found.
    2. Start with the initial array element.
    3. repeat while the target is not found and there are more array elements
        4. if the current element matches the target
                5. Set a flag to indicate that the target has been found.
        else
                6. Advance to the next array element.
    7. if the target was found
                8. Return the target index as the search result.
        else
                9. Return -1 as the search result.

    Create a user-defined function with prototype **`int linear_search(const int arr[ ], int target, int n);`** in your program to search the target element.

Space for Program **Output ▼**

Space for Program **Output ▼**

13. Write a program to copy the distinct elements of an int type array to another int type array. For example, if the input array is 4 7 7 3 2 5 5 then the output array will be 4 7 3 2 5.

Space for Program **Output ▼**

14. A barcode scanner for Universal Product Codes (UPCs) verifies the 12-digit code scanned by comparing the code's last digit (called a check digit) to its own computation of the check digit from the first 11 digits as follows:

    (1) Calculate the sum of the digits in the odd-numbered positions (the first, third, ..., eleventh digits) and multiply this sum by 3.

    (2) Calculate the sum of the digits in the even-numbered positions (the second, fourth, ..., tenth digits) and add this to the previous result.

    (3) If the last digit of the result from step 2 is 0, then 0 is the check digit. Otherwise, subtract the last digit from 10 to calculate the check digit.

    (4) If the check digit matches the final digit of the 12-digit UPC, the UPC is assumed correct.

    Write a program that prompts the user to enter the 12 digits of a barcode separated by spaces. The program should store the digits in an integer array, calculate the check digit, and compare it to the final barcode digit. If the digits match, output the barcode with the message "validated". If not, output the barcode with the message "error in barcode". Also, output with labels the results from steps 1 and 2 of the check-digit calculations. Note that the "first" digit of the barcode will be stored in element 0 of the array. Try your program on the following barcodes, three of which are valid. For the first barcode, the result from step 2 is 79 (0 + 9 + 0 + 8 + 4 + 0) * 3 + (7 + 4 + 0 + 0 + 5).

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

Space for Program **Output ▼**

Space for Program **Output ▼**

15. Write a program to grade an n-question multiple-choice exam (for n between 5 and 50) and provide feedback about the most frequently missed questions. Your program will take data from the file **`examdat.txt`**. The first line of the file contains the number of questions on the exam followed by a space and then an n -character string of the correct answers. Write a function fgetAnswers that inputs the answers from an open input file. Each of the lines that follow contain an integer student ID followed by a space and then that student's answers. Function fgetAnswers can also be called to input a student's answers. Your program is to produce an output file, **`report.txt`**, containing the answer key, each student's ID and each student's score as a percentage, and then information about how many students missed each question. Here are short sample input and output files.

examdat.txt

```
5 dbbac
111 dabac
102 dcbdc
251 dbbac
```

report.txt

```
                Exam Report
Question      1   2   3   4  5
Answer        d   b   b   a  c

ID        Score(%)
111        80
102        60
251        100

Question      1   2   3   4  5
Missed by     0   2   0   1  0
```

Space for Program **Output ▼**

Space for Program **Output ▼**

WEEK-END ASSIGNMENT-07

Working with pointers, referencing a variable through a pointer and accessing the contents of a memory cell through a pointer variable that stores its address (i.e. indirect reference).

Assignment Objectives:

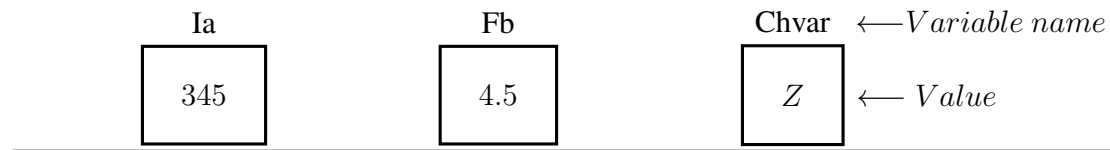
To learn about pointers, referencing, indirect referencing and how to return function results through a function's parameters (input parameters, input/output parameters, output parameters). Also to understand the differences between call-by value & call-by-reference.

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. For the given structure below, declare the variable type, and print their values as well as addresses;

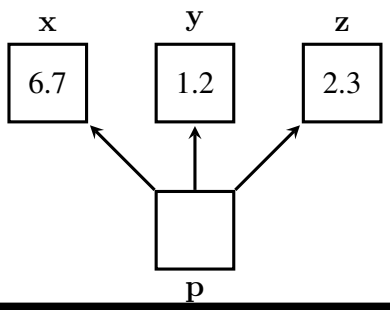


Space for Program ▼ | **Output ▼**

2. Declare two integer variable and assign values to them, and print their addresses. Additionally, Swap the contents of the variables and print their addresses after swap. State whether the addresses before and after are equal or not.

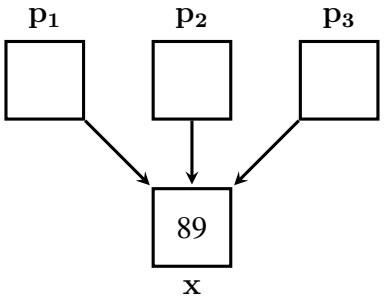
Space for Program ▼ | **Output ▼**

3. Write the C statement to declare and initialize the pointer variable,**p**, for the given structure and display the values of **x**, **y** and **z** with the help of **p**.



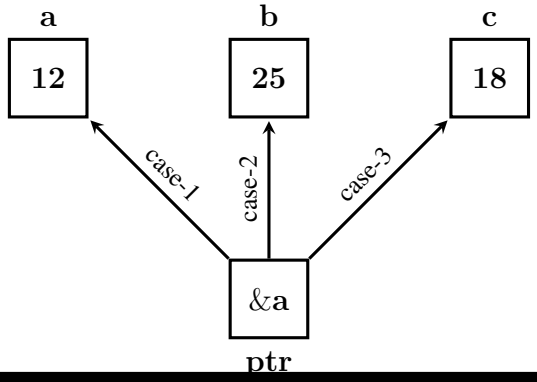
Space for Program ▼ **Output ▼**

4. Write the C statement to declare and initialize the pointer variables p_1, p_2 and p_3 for the given structure and display the value of **x** from p_1. Also update the value of **x** to 100 using pointer p_3.



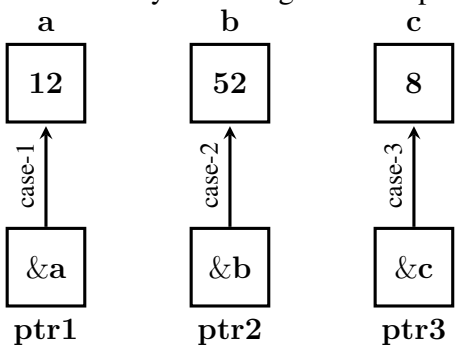
Space for Program ▼ **Output ▼**

5. Write the C statement to declare and initialize the pointer variable for the given structure and update the values of **a**, **b** and **c** to be incremented by 10 through the pointer variable.



Space for Program ▼ **Output ▼**

6. Write the C statement to declare and initialize the pointer variable for the given structure and update the values of **a**, **b** and **c** to be incremented by 10 through their respective pointers.



Space for Program ▼ **Output ▼**

7. Two pointers are pointing to different variables. Write the C statement to find the greater between **a**, and **b** using pointer manipulation.

Space for Program ▼ **Output ▼**

8. Create a program to display the address and value of each element of the given integer array **a**. Also perform a close observation on the format of the address and the change of address from index **0** to the last index of the array.

0	1	2	3	4	5	6	7	8	9
0	10	20	30	40	50	60	70	80	90
&**a**[0]	&**a**[1]	&**a**[2]	&**a**[3]	&**a**[4]	&**a**[5]	&**a**[6]	&**a**[7]	&**a**[8]	&**a**[9]

Space for Program ▼ **Output ▼**

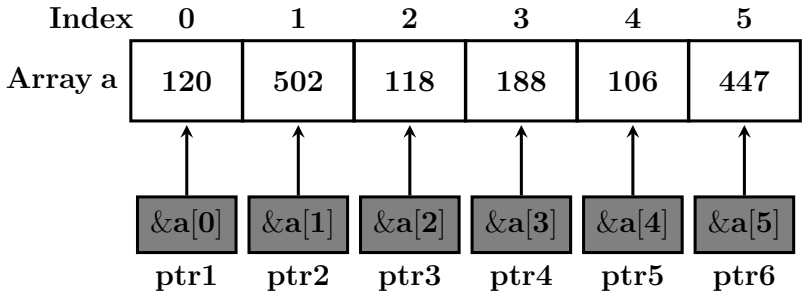
9. Declare the two arrays to hold the values as shown in the given rectangular boxes. Write the equivalent C statement to print their values and addresses through pointer (**`Hint:`** an array name is a pointer to the first element in the array).

10	13	20	33	44

10.2	13.3	20.0	33.3	45.3	89.9

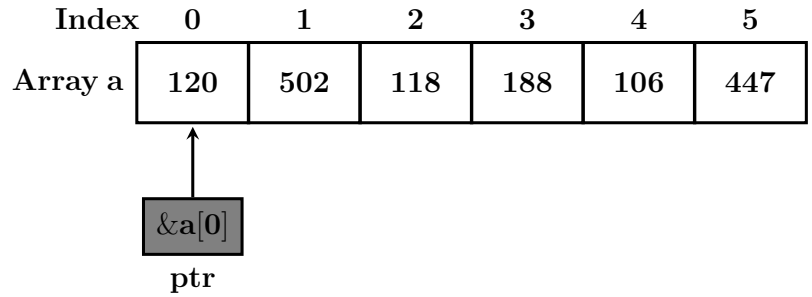
Space for Program ▼ **Output ▼**

10. Write the C statement to declare and initialize the pointer variable for the given structure and display the array content using pointer.



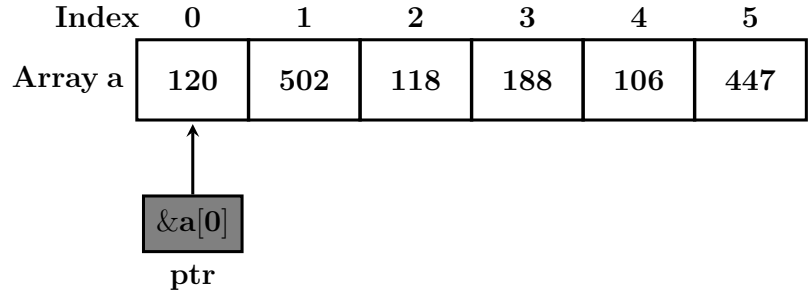
Space for Program **Output ▼**

11. Write the C statement to declare and initialize the pointer variable for the given structure and display the array content using pointer.



Space for Program **Output ▼**

12. As array name is a pointer, so modify the assignment **ptr=a** rather **ptr=&a[0]**. Write the C statement to declare and initialize the pointer variable for the given structure and display the array content using pointer.



Space for Program **Output ▼**

13. Trace the execution of the following fragment at **`line -1`**.

```
int m = 10, n = 5;
int *mp, *np;
mp = &m;
np = &n;
*mp = *mp + *np;
*np = *mp - *np;
printf("%d %d\n%d %d\n", m, *mp, n, *np); /*
    line-1 */
```

Output▼

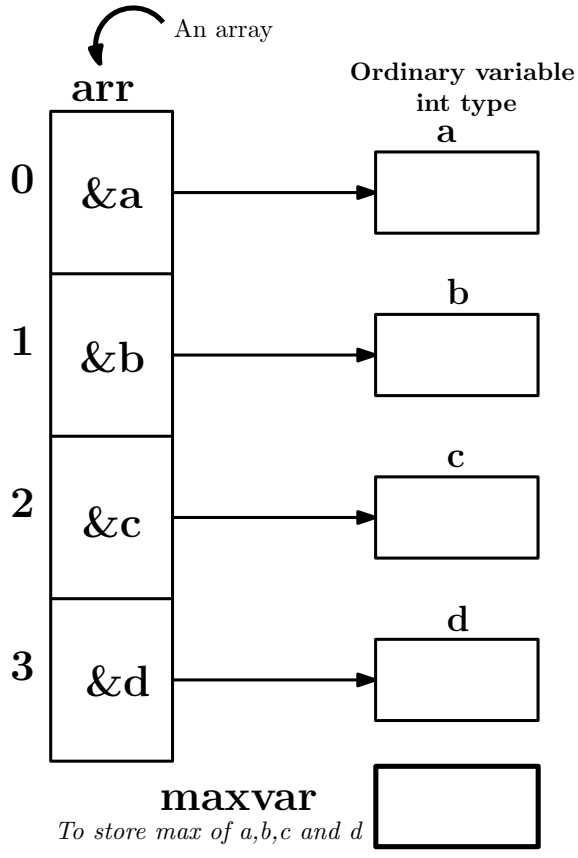
14. Given the declarations;

```
int m = 25, n = 77;
char c = '*';
int *itemp;
/* describe the errors in each of the
following statements. */
m = &n;
itemp = m;
*itemp = c;
*itemp = &c;
```

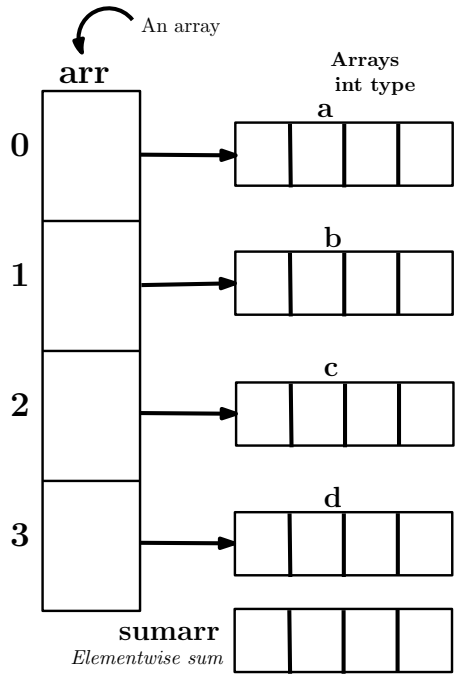
Output▼

15. Simulate the following structure in C to store **55** in **a**, **105** in **b**, **89** in **c** and **68** in **d** using their respective pointers. Additionally find the maximum among **a**, **b**, **c** and **d** through pointer manipulation. Finally Store the maximum to the required variable and display the maximum.

C simulation▼



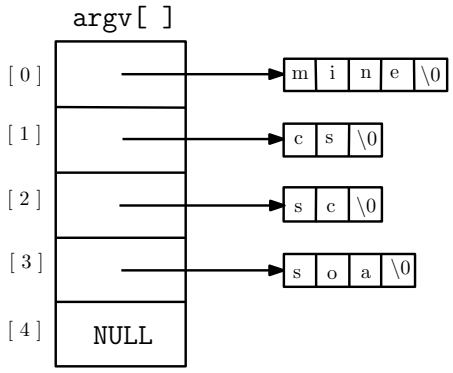
16. Simulate the following structure in C to find the element sum of the given arrays **a**, **b**, **c** and **d** into **sumarray** using their respective pointers. The 1-D arrays must be read/scanned through the pointers.



C simulation▼

Test case: Input & Output▼

17. An argument array is an array of pointers to strings. The end of the array is marked by an entry containing a `NULL` pointer as shown in the figure. Write a C Simulation to implement the following figure and manipuate the character array to hold all capital case letters using pointer. Finally display the strings.



C simulation▼

Test case: Input & Output▼

18. Consider the following figures 1, 2 and 3 to manipulate the ordinary variables, integer arrays and strings through pointers. There exist no names associated with the variables, arrays and strings. State the method to allocate memory for the pointers to manipulate the desired variables.

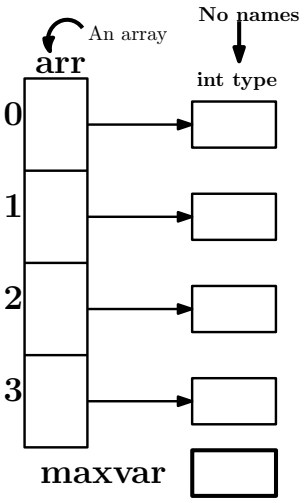


Figure-1

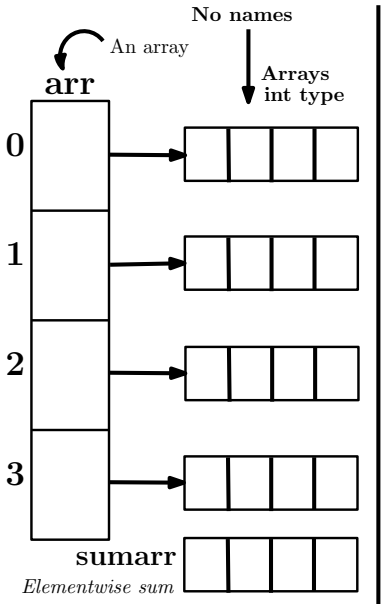


Figure-2

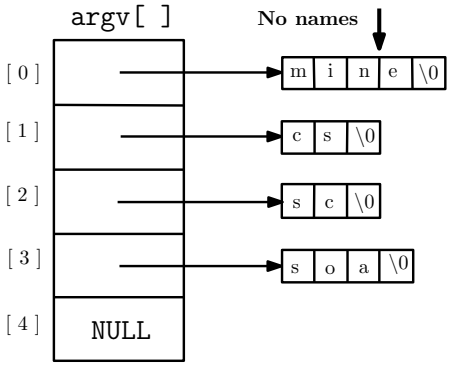
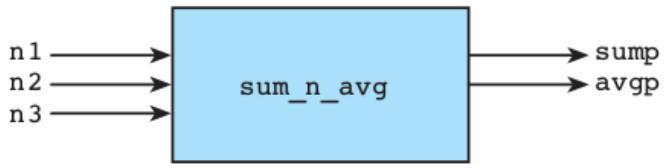


Figure-3

Answer ▼

19. Write a prototype for a function **sum_n_avg** that has three type double input parameters and two output parameters. The function computes the sum and the average of its three input arguments and relays its results through two output parameters.



Output▼

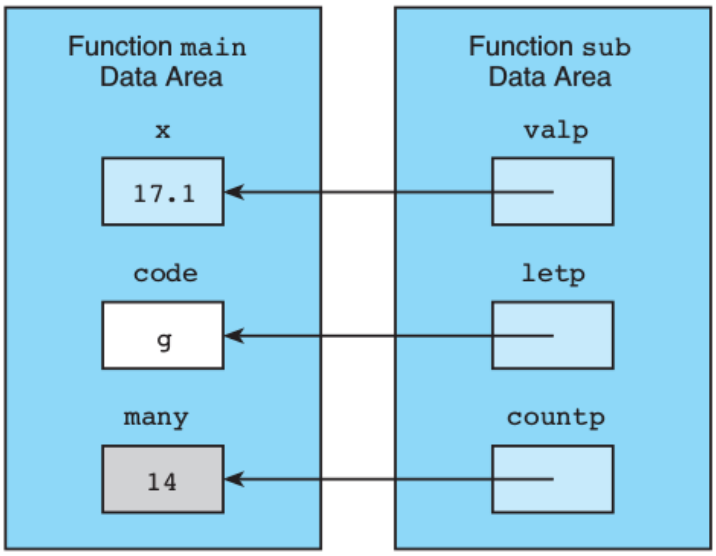
20. The following code fragment is from a function preparing to call **sum_n_avg** (see question-19). Complete the function call statement.

```
double one, two, three, sum_of_3, avg_of_3;
printf("Enter three numbers> ");
scanf("%lf%lf%lf", &one, &two, &three);
sum_n_avg(_____________________________________________________________);
. . .
```

Define the function **sum_n_avg** whose prototype you wrote in question-19.

Space for Program	Output ▼

21. Given the memory setup shown, fill in the chart by indicating the data type and value of each reference as well as the name of the function in which the reference would be legal. Describe pointer values by



referring to cell attributes. For example, the value of **`valp`** would be "pointer to color-shaded cell", and the value of **`&many`** would be "pointer to gray-shaded cell".

Reference	Where Legal	Data Type	Value
`valp`	`sub`	`double *`	pointer to color-shaded cell
`&many`			
`code`			
`&code`			
`countp`			
`*countp`			
`*valp`			
`letp`			
`&x`			

22. Write a program to use the idea of **multiple calls to a function with input/output parameters** to sort 6 integer numbers in ascending order without using any sorting algorithms. The prototype of the function to be used in your program to sort the numbers is given as **`void arrange(int *, int *);`** and also draw the data areas of calling function and **`arragne()`** function for the first function call **`arrange(....)`**.

Sample Run

```
printf("Enter SIX numbers separated by blanks> ");
12 3 56 8 20 654

/* Displays results */
printf("The numbers in ascending order are: %d %d %d %d %d %d\n", n1,
   n2, n3, n4, n5, n6);
3 8 12 20 56 654
```

Space for Program ▼ **Output ▼**

23. Show the table of values for x , y , and z that is the output displayed by the following program.

```
#include <stdio.h>
void sum(int a, int b, int *cp);
int main(void){
 int x, y, z;
 x = 7; y = 2;
 printf("x y z\n\n");
 sum(x, y, &z);
 printf("%4d%4d%4d\n", x, y, z);
 sum(y, x, &z);
 printf("%4d%4d%4d\n", x, y, z);
 sum(z, y, &x);
 printf("%4d%4d%4d\n", x, y, z);
 sum(z, z, &x);
 printf("%4d%4d%4d\n", x, y, z);
 sum(y, y, &y);
 printf("%4d%4d%4d\n", x, y, z);
 return (0);
}
```

```
void sum(int a, int b, int *cp){
  *cp = a + b;
}
```

Output▼

24. (a) What values of x and y are displayed by this program? (Hint: Sketch the data areas of **`main`** , **`trouble`** , and **`double_trouble`** as the program executes.)

```
void double_trouble(int *p, int y);
void trouble(int *x, int *y);
int main(void){
  int x, y;
  trouble(&x, &y);
  printf("x = %d, y = %d\n", x, y);
  return (0);
}
void double_trouble(int *p, int y){
  int x;
  x = 10;
  *p = 2 * x - y;
}
void trouble(int *x, int *y){
  double_trouble(x, 7);
  double_trouble(y, *x);
}
```

Output▼

(b) Classify each formal parameter of **`double_trouble`** and **`trouble`** as input, output, or input/output.

Formal parameter classification ▼

25. A finite state machine (FSM) consists of a set of states, a set of transitions, and a string of input data. In the FSM of Figure 1, the named ovals represent states, and the arrows connecting the states represent transitions. The FSM is designed to recognize a list of **C identifiers** and **nonnegative integers**, assuming that the items are ended by one or more blanks and that a period marks the end of all the data. The following table traces how the diagrammed machine would process a string composed of one blank, the digits 9 and 5, two blanks, the letter K, the digit 9, one blank, and a period. The machine begins in the start state.

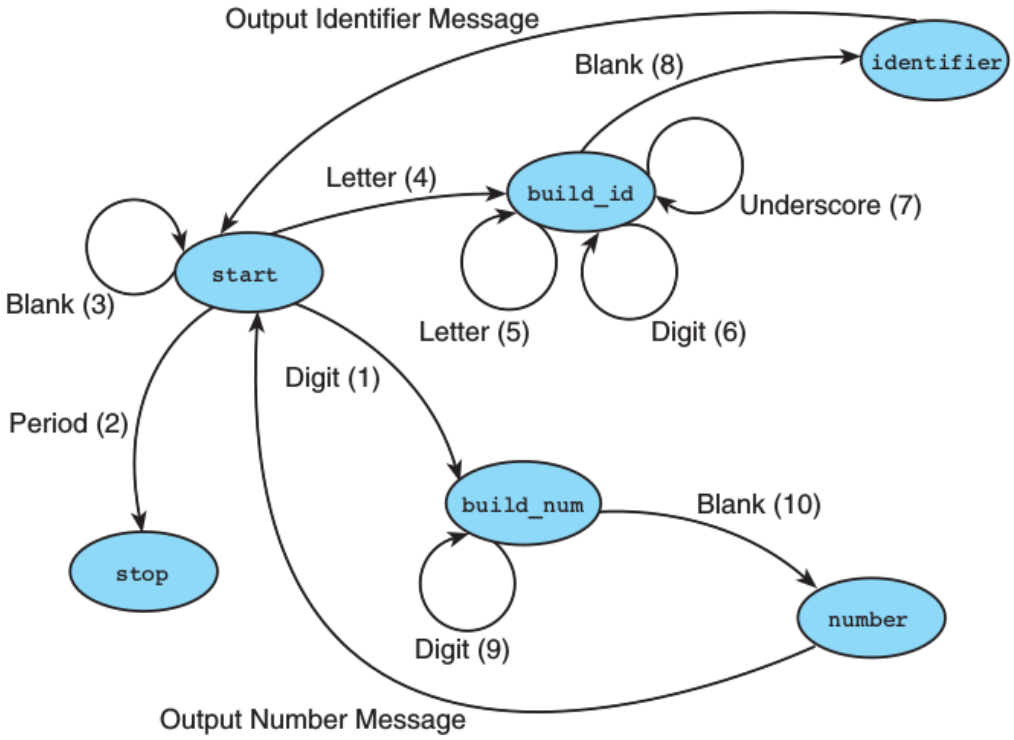


Figure 1: Finite State Machine for Numbers and Indentifiers

Figure 2: **FSM tracing for 95 and K9**

State	Next Character	Transition
start	' '	3
start	'9'	1
build_num	'5'	9
build_num	' '	10
number		Output number message
start	' '	3
start	'K'	4
build_id	'9'	6
build_id	' '	8
identifier		Output identifier message
start	'.'	2
stop		

Write a program that uses an enumerated type to represent the names of the states. Your program should process a correctly formatted line of data, identifying each data item. Here is a sample of correct input and output.

```
Input :

rate R2D2 48 2 time 555666

Output :

rate - Identifier
R2D2 - Identifier
48 - Number
2 - Number
time - Identifier
555666 - Number
```

Use the following code fragment in **main** , and design function **transition** to return the next state for all the numbered transitions of the finite state machine. If you include the header file **ctype.h**, you can use the library function **isdigit** which returns 1 if called with a digit character, 0 otherwise. Similarly, the function **isalpha** checks whether a character is a let- ter. When your program correctly models the behavior of the **FSM** shown, extend the **FSM** and your program to allow optional signs and optional fractional parts (i.e., a decimal point followed by zero or more digits) in numbers.

```
current_state = start;
do {
   if (current_state == identifier) {
        printf(" - Identifier\n");
        current_state = start;
   } else if (current_state == number) {
        printf(" - Number\n");
        current_state = start;
   }
   scanf("%c", &transition_char);
   if (transition_char != ' ')
        printf("%c", transition_char);
        current_state = transition(current_state, transition_char);
   } while (current_state != stop);
```

FSM Inplementation ▼

FSM Inplementation ▼

FSM Inplementation ▼

WEEK-END ASSIGNMENT-08

Experiment with programs, processes, memory allocation and manipulation for processes in UNIX.

Assignment Objectives:

Students will be able to differentiate programs and processes, also able to learn how to create processes and able to explore the implication of process inheritance.

Instruction to Students (If any):

Students are required to write his/her own program/output 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. Construct the process tree diagram and also find the number of processes along with output of the following code snippet.

```
int main(void) {
    fork();
    fork();
    fork();
    printf("ITER\n");
    printf("ITER\n");
    return 0;
}
/* Any formula can be
    devised for number
     of processes
    created here?
If so, state.*/
```

Process tree, # of processes and output

2. Construct the process tree diagram and also find the number of processes along with output of the following code snippet.

```
int main(void){
  printf("hello\n");
  fork();
  printf("hello\n");
  fork();
  printf("hello\n");
  fork();
  printf("hello\n");
  return 0;
}
/* Any formula for
    nuber of outputs?
If so, state.*/
```

Process tree, # of processes and output

3. Run the following code on your machine and write the output. Suggest a way to avoid the mismatch of machine output w.r.t. dry run output.

```
#include<stdio.h>
#include<unistd.h>
int main(void)
{
   printf("A");
   fork();
   printf("P\n");
   return 0;
}
```

Output, Reason and Suggestion

4. Draw the process tree and write the output of the following code snippet.

```
int main()
{
   fork() && fork();
   printf("Able to\n");
   return 0;
}
```

Process tree and output

5. Draw the process tree and write the output of the following code snippet.

```
int main()
{
   fork();
   fork() && fork();
   fork();
   printf("Got!!!\n");
   return 0;
}
```

Process tree and output

6. Draw the process tree and write the output of the following code snippet.

```
int main()
{
   fork();
   fork() + fork();
   fork();
   printf("doing!\n");
   return 0;
}
```

Process tree and output

7. Draw the process tree and write the output of the following code snippet.

```
int main(){
   fork();
   fork() || fork();
   fork();
   printf("Really!!!\n");
   return 0;
}
```

Process tree | **Output**

Remark

8. Draw the process tree and write the output of the following code snippet.

```
int main(){
 fork();
 fork()&& fork()||fork();
 fork();
 printf("guess\n");
 return 0;
}
```

Process tree | **Output**

Remark

9. Draw the process tree and write the output of the following code snippet.

```
int main(){
 fork()&&fork();
 fork()||fork();
 printf("Hi\n");
 return 0;
}
```

Process tree | **Output**

Remark

10. Construct the process tree diagram and also find the number of processes along with output of the following code snippet.

```
int main(){
 int pid,pid2;
 pid=fork();
 if(pid){
   pid2=fork();
   printf("I\n");
 }
 else{
     printf("C\n");
     pid2=fork();
 }
 return 0;
}
```

Process tree, # of processes and output		

11. Construct the process tree diagram and also find the number of processes along with output of the following code snippet.

```
int
main(void){
 pid_t childpid;
 int i, n=3;
 for(i=1;i<n;i++){
  childpid=fork();
  if(childpid==-1)
     break;
 }
 printf("i:%d\n",i);
 return 0;
}
```

Process tree, # of processes and output		

12. Draw the process tree because of the following code snippet and state number of times **x=0** as well as **x≠0**will be displayed.

```
pid_t
add(pid_t a, pid_t b){
  return a+b;
}
int main(void){
 pid_t x=10;
 printf("%d\n",x);
 x=add(fork(),fork());
 printf("%d\n",x);
 return 0;}
```

Process tree	# of x=0	# of x≠0

13. Determine the total number of displayed for the given code snippet.

```
int main(void){
 int x[]={10,20,fork(),fork()+fork()};
 int len=sizeof(x)/sizeof(int);
 for(int i=0;i<len;i++)
   fprintf(stderr," %d ",x[i]);
 printf("\n");
 return 0;
}
```

# of display	# of Processes

14. Determine the number of process(s) will be created when the below program becomes process and also write the output.

```
void show(){
  if(fork()==0)
   printf("1\n");
  if(fork()==0)
   printf("2\n");
 if(fork()==0)
   printf("3\n");
}
int main(void){
 show();
 return 0;
}
```

# of processes	Output

15. Draw the process tree of the following code snippet. Also give a count of processes and the output of the following code. Can the code segment generate fan of processes.

```
int main(void){
 if(fork()==0)
  printf("1\n");
else if(fork()==0)
  printf("2\n");
else if(fork()==0)
  printf("3\n");
else if(fork()==0)
  printf("4\n");
else
  printf("5\n");
return 0;
}
```

Process tree, # of processes and output		

16. Find the output of the code segment showing the corresponding process tree.

```
int main(){
 pid_t p1,p2;
 p2=0;
 p1=fork();
 if (p1 == 0)
    p2 = fork();
 if (p2 > 0)
    fork();
 printf("done\n");
return 0;
}
```

Process tree	Output

17. Find the number of direct children to the main process, the total number of processes and the output.

```
int main(){pid_t c1=1,c2=1;
 c1=fork();
 if(c1!=0)
    c2=fork();
 if(c2==0){
    fork();printf("1\n");
  }
return 0;}
```

# of direct children to main process	Total processes	Output

18. Find the output of the code segment.

```
int main(){
  struct stud s1={1,20};
  pid_t pid=fork();
  if(pid==0){
     struct stud s1={2,30};
     printf("%d %d\n",s1.r,s1.m);
     return 0;
  }
  else{
    sleep(10);
    printf("%d %d\n",s1.r,s1.m);
    return 0;
  }
}
```

```
struct stud{
  int r;
  int m;
};
```

Output

19. Find the output of the code segment showing the corresponding process tree.

```
int main(){
  if(fork()){
    if(!fork()){
      fork();
      printf("S ");
    }
    else{
      printf("T ");
    }
 }
 else{
      printf("D ");
 }
 printf("A ");
 return 0;
}
```

Process tree **Output**

20. Calculate the number of processes the following code snippet will generate.

```
int main(){int i;
  for(i=0;i<12;i++){
     if(i%3==0){
          fork();
     }
 }
 return 0;
}
```

Process tree **Output**

21. State the possible values of x for the given code snippet;

```
int x;
int a[2]={10,20};
x=5+a[fork() || fork()];
printf("%d ",x);
```

Process tree **Output**

22. Suppose four user-defined exit handlers X, Y, P, and Q are installed in the order X then Y then P then Q using atexit() function in a C program. Exit hadler X is designed to display 1, Y is designed to display 2, P is designed to display 3, and Q to display 4. State the order of their display, when the program is going to terminate after calling return 0/exit(0).

(A) 4, 3, 2 1
(B) 1,2,3,4
(C) 1, 2, 4, 3
(D) none

Choice

23. You know that the **ps** utility in UNIX reports a snapshot of the current processes. Determine the state code of the given program, that became a process.

```
int main(void){
 fprintf(stderr,"PID=%ld\n",(long)getpid());
 while(1);
 return 0;
}
```

(A) R
(B) S
(C) T
(D) Z

Choice

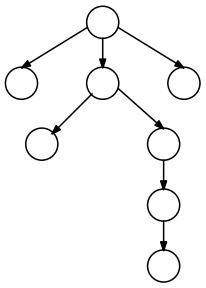
24. Find the process state code of the given program, that became a process using the Unix utility **ps**. As you know **ps** displays information about a selection of the active processes.

```
int main(void){
 fprintf(stderr,"PID=%ld\n",(long)getpid());
 while(1)
     sleep(1);
 return 0;
}
```

(A) R
(B) S
(C) T
(D) Z

Choice

25. Develop a C code to create the following process tree. Display the process ID, parent ID and return value of `fork()` for each process.



OBSERVARIONS:

- Use ps utility to verify the is-a-parent relationship?
- Are ypu gettig any orphan process case?
- Are you getting any ZOMBIE case?

Figure 1: Process tree

Code here

26. Create two different user-defined functions to generate the following process hierarchy shown in **Figure-(a)** and **Figure-(b)**. Finally all the processes display their process ID and parent ID.

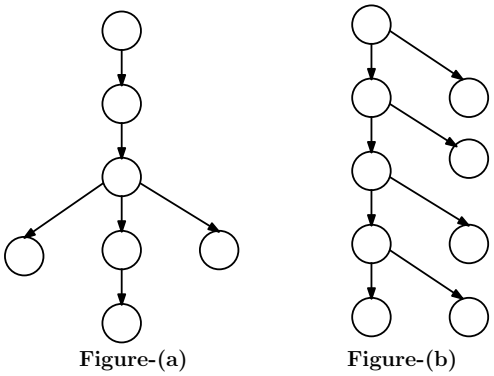


Figure 2: Process tree

Code here

Code here

27. What output will be at Line X and Line Y?

```
#define SIZE 5
int nums[SIZE] = { 0,1,2,3,4 } ;
int main(){
 int i;
 pid t pid;
 pid = fork();
 if(pid == 0){
    for (i = 0; i < SIZE; i++) {
       nums[i] *= nums[i] *-i;
       printf("CHILD:%d ",nums[i]);  /* LINE X */
    }
 }
 else if (pid > 0) {
    wait(NULL);
    for (i = 0; i < SIZE; i++)
       printf("PARENT: %d ",nums[i]);  /* LINE Y */
    }
  return 0;
}
```

Output

28. The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8,

    Write a C program using the fork() system call that generates the Fibonacci sequence in the child process. The number of the sequence will be provided in the command line. For example, if 5 is provided, the first five numbers in the Fibonacci sequence will be output by the child.

Code here

29. Write a **`MulThree.c`** program to multiply three numbers and display the output. Now write another C program **`PracticeExecl.c`**, which will fork a child process and the child process will execute the file **`MulThree.c`** and generate the output. The parent process will wait till the termination of the child and the parent process will print the process ID and exit status of the child.

Code here

30. You know the usages of the command **grep**. Implement the working of **grep -n pattern filename** in a child process forked from the parent process using **execl** system call. The parent process will wait till the termination of the child and the parent process will print the process ID and exit status of the child.

Code here

31. Implement the above question using **`execv`**, **`execlp`**, **`execvp`**, **`execle`**, **`execve`** system calls.

Code here for `execv`

Code here for `execlp`

Code here for `execvp`

Code here for `execve`

Code here for `execle`