

# **Lab Course I**

## Exercise 1

Start Date

/ /



**To demonstrate the use of data types, simple operators and expressions**



You should read following topics before starting this exercise

1. Different basic data types in C and rules of declaring variables in C
2. Different operators and operator symbols in C
3. How to construct expressions in C, operator precedence
4. Problem solving steps- writing algorithms and flowcharts



### 1. Data type Table

Data	Data Format	C Data Type	C Variable declaration	Input Statement	Output statement
quantity month credit-card number	Numeric	int Short int long int	int quantity; short month; long ccno;	scanf("%d",&quantity); scanf("%d",&month); scanf("%ld", &ccno);	printf("The quantity is %d", quantity); printf("The credit card number is %ld, ccno);
price $\pi$	real	float double	float price; const double pi=3.141593;	scanf("%f",&price);	printf("The price is %5.2f", price);
grade	character		char grade;	scanf("%c",&grade)	printf("The grade is %c",grade);

### 2. Expression Examples

Expression	C expression
Increment by a 3	a = a + 3
Decrement b by 1	b = b-1 or b--
$2a^2 + 5b/2$	2*a*a + 5*b/2
$7/13(x-5)$	(float)7/13*(x-5)
5% of 56	(float)5/100*56
n is between 12 to 70	n>=12 && n<=70
$\pi^2 h$	Pi*r*r*h
n is not divisible by 7	n % 7 != 0
n is even	n%2== 0
ch is an alphabet	ch>='A' && ch<='Z'    ch>='a' && ch<='z'

Note: The operators in the above expressions will be executed according to precedence and associativity rules of operators.

3. Sample program- to calculate and print simple interest after accepting principal sum, number of years and rate of interest.

## Program development steps

Step 1 : Writing the Algorithm	Step 2 : Draw the flowchart	Step 3 : Writing Program
<ol style="list-style-type: none"> <li>1. Start</li> <li>2. Accept principal sum, rate of interest and number of years</li> <li>3. Compute Simple interest</li> <li>4. Output Simple Interest</li> <li>5. Stop</li> </ol>	<pre> graph TD     Start([start]) --&gt; Read[/Read ,principal sum, rate and no of years/]     Read --&gt; Compute[Compute Simple interest]     Compute --&gt; Print[/Print Simple Interest/]     Print --&gt; Stop([stop]) </pre>	<pre> /* Program to calculate simple interest */ #include &lt;stdio.h&gt; main( ) { /* variable declarations */   float amount, rateOfInterest, simpleInterest;   int noOfYears;   /* prompting and accepting input */   printf("Give the Principal Sum");   scanf("%f",&amp;amount);   printf("Give the Rate of Interest");   scanf("%f",&amp;rateOfInterest);   printf("Give the Number of years");   scanf("%d",&amp;noOfYears);    /* Compute the simple Interest*/   simpleInterest=amount*noOfYears*rateOfInterest / 100;    /* Print the result*/   printf("The simple Interest on amount %7.2f for %d years at the rate %4.2f is %6.2f", amount, noOfYears, rateOfInterest, simpleInterest); } </pre>



1. Type the sample program given above. Execute it for the different values as given below and fill the last column from the output given by the program.  
Follow the following guidelines
  - a. At \$ prompt type vi followed by filename. The filename should have .c as extension for example  
\$vi pnr.c
  - b. Type the sample program given above using vi commands and save it  
Compile the program using cc compiler available in Linux  
\$cc pnr.c  
It will give errors if any or it will give back the \$ prompt if there are no errors  
A executable file a.out is created by the compiler in current directory. The program can be executed by typing name of the file as follows giving the path.  
\$ ./a.out  
Alternatively the executable file can be given name by using -o option while compiling as follows  
\$cc pnr.c -o pnrexec  
\$./pnrexec  
The executable file by specified name will be created. Note that you have to specify the path of pnrexec as ./pnrexec , i. e., pnrexec in current (. Stands for current directory) directory otherwise it looks for program by that name in the path specified for executable programs

Sr. No	Principal sum	No of years	Rate of interest	Simple Interest
1	2000	3	_____	
2	4500	_____	4.5	
3	_____	6	8.3	

2. If you have not typed the program correctly, i.e., if there are syntactical errors in the program, compiler will pinpoint the errors committed and are called compile-time errors. C compiler gives line no along with error messages when it detects grammatical or syntactical errors in the program. These messages are not so straightforward and you may find it difficult to identify the error. You may miss a semicolon at the end of a statement and the compiler points out error in the next statement. You may miss just a closing '\*' of a comment and it will show errors in several statements following it.
- Another type of error which is quite common is the run-time or execution error. You are able to compile the program successfully but you get run-time messages or garbage output when you execute the program.
- Modify the above program to introduce the following changes, compile, write the error messages along with line numbers, remove the error execute and indicate the type of error whether it was compile-time or execution time error.

Modified line	Error messages and line numbers	Type of error
/* Program to calculate simple interest		
int noofYears;		
scanf("%f",&amount)		
scanf("%f", amount);		
scanf("%d", noOfYears);		

Signature of the instructor

Date

 /  / 


**Set A . Apply all the three program development steps for the following examples.**

1. Accept dimensions of a cylinder and print the surface area and volume (Hint: surface area =  $2\pi r^2 + 2\pi rh$ , volume =  $\pi r^2 h$ )
2. Accept temperatures in Fahrenheit (F) and print it in Celsius (C) and Kelvin (K) (Hint:  $C = 5/9(F - 32)$ ,  $K = C + 273.15$ )
3. Accept initial velocity (u), acceleration (a) and time (t). Print the final velocity (v) and the distance (s) travelled. (Hint:  $v = u + at$ ,  $s = u + at^2$ )
4. Accept inner and outer radius of a ring and print the perimeter and area of the ring (Hint: perimeter =  $2\pi(a+b)$ , area =  $\pi(a^2 - b^2)$ )
5. Accept two numbers and print arithmetic and harmonic mean of the two numbers (Hint:  $AM = (a+b)/2$ ,  $HM = ab/(a+b)$ )
6. Accept three dimensions length (l), breadth (b) and height (h) of a cuboid and print surface area and volume (Hint : surface area =  $2(lb + lh + bh)$ , volume =  $lbh$ )
7. Accept a character from the keyboard and display its previous and next character in order.  
Ex. If the character entered is 'd', display "The previous character is c", "The next character is e".
8. Accept a character from the user and display its ASCII value.

Signature of the instructor

Date

 /  / 

**Set B . Apply all the three program development steps for the following examples.**

1. Accept the x and y coordinates of two points and compute the distance between the two points.
2. Accept two integers from the user and interchange them. Display the interchanged numbers.
3. A cashier has currency notes of denomination 1, 5 and 10. Accept the amount to be withdrawn from the user and print the total number of currency notes of each denomination the cashier will have to give.

Signature of the instructor

Date

 /  / 

**Set C. Write a program to solve the following problems**

1. Consider a room having one door and two windows both of the same size. Accept dimensions of the room, door and window. Print the area to be painted (interior walls) and area to be whitewashed (roof).
2. The basic salary of an employee is decided at the time of employment, which may be different for different employees. Apart from basic, employee gets 10% of basic as house rent, 30% of basic as dearness allowance. A professional tax of 5% of basic is deducted from salary. Accept the employee id and basic salary for an employee and output the take home salary of the employee.

Signature of the instructor

Date

 /  / 

**Assignment Evaluation**

**Signature**

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

## Exercise 2

Start Date

 /  / 


To demonstrate use of decision making statements such as if and if-else.



You should read following topics before starting this exercise

1. Different types of decision-making statements available in C.
2. Syntax for these statements.

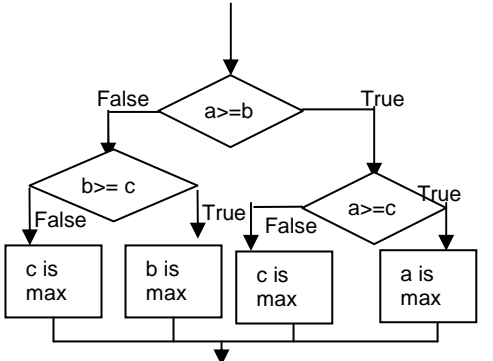


During problem solving, we come across situations when we have to choose one of the alternative paths depending upon the result of some condition. Condition is an expression evaluating to true or false. This is known as the Branching or decision-making statement. Several forms of If and else constructs are used in C to support decision-making.

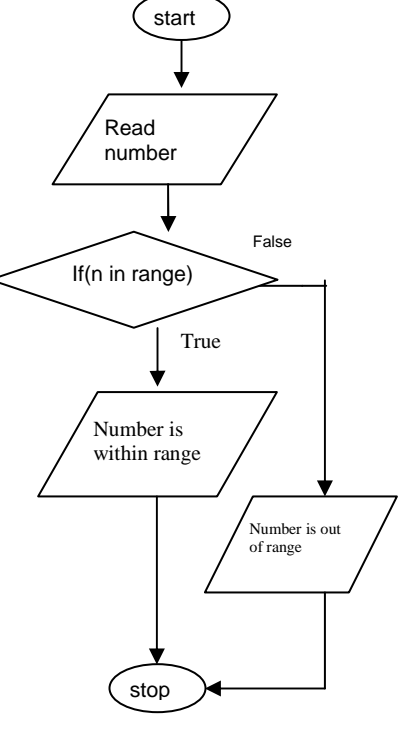
- 1) if statements
- 2) if – else
- 3) Nested if

Note: If there are more than one statement in the if or else part, they have to be enclosed in { } braces

Sr. No	Statement Syntax	Flowchart	Example
1.	<b>if statement</b>  if (condition) { statement; }	<pre> graph TD     Start(( )) --&gt; Cond{If condition?}     Cond -- True --&gt; Stmt[statement]     Cond -- False --&gt; NewStmt[New statement]     Stmt --&gt; NewStmt           </pre>	<pre> if( n &gt; 0)     printf("Number     is positive");           </pre>
2.	<b>if - else statement</b>  if (condition) { statement; } else { statement; }	<pre> graph TD     Start(( )) --&gt; Cond{If condition?}     Cond -- True --&gt; Stmt1[statement]     Cond -- False --&gt; Stmt2[statement]     Stmt1 --&gt; NewStmt[New statement]     Stmt2 --&gt; NewStmt           </pre>	<pre> if( n % 2 == 0)     printf("Even"); else     printf("Odd");           </pre>

3.	<b>Nested if</b>  <pre> if (condition) {     if (condition)     { statement;} else { statement;} } else {     if (condition)     { statement; } else { statement; } } </pre>	 <pre> graph TD     Start(( )) --&gt; D1{a &gt;= b}     D1 -- True --&gt; AMax[a is max]     D1 -- False --&gt; D2{b &gt;= c}     D2 -- True --&gt; BMax[b is max]     D2 -- False --&gt; CMax[c is max]     AMax --&gt; End(( ))     BMax --&gt; End     CMax --&gt; End </pre>	<pre> If ( a &gt;= b) { if ( a &gt;= c)     printf(" %d is maximum",a);   else     printf(" %d is maximum",c); } else {   if ( b &gt;= c)     printf(" %d is maximum",b);   else     printf(" %d is maximum",c); } </pre>
----	--	--	---

4. Sample program- to check whether a number is within range.

Step 1: Writing the Algorithm	Step 2 : Draw the flowchart	Step 3 : Writing Program
<ol style="list-style-type: none"> <li>Start</li> <li>Accept the number</li> <li>Check if number is within range</li> <li>if true     print "Number is within range "     otherwise     print "number is out of range".</li> <li>Stop</li> </ol>	 <pre> graph TD     Start([start]) --&gt; Read[/Read number/]     Read --&gt; D1{If(n in range)}     D1 -- True --&gt; WRange[/Number is within range/]     D1 -- False --&gt; ORange[/Number is out of range/]     WRange --&gt; Stop([stop])     ORange --&gt; Stop </pre>	<pre> /* Program to check range */  #include &lt;stdio.h&gt; main( ) { /* variable declarations */   int n;   int llimit=50, ulimit = 100;   /* prompting and accepting input */   printf("Enter the number");   scanf("%d",&amp;n);   if(n&gt;=llimit &amp;&amp; n &lt;= ulimit)     printf("Number is within range");   else     printf("Number is out of range"); } </pre>



1. Execute the following program for five different values and fill in the adjoining table

<pre>main() {   int n;   printf("Enter no.");   scanf("%d",&amp;n);   if(n%__==0)     printf("divisible");   else     printf("not divisible"); }</pre>	n	output

2. Type the above sample program 4 and execute it for the following values.

n	Output message
50	
100	
65	
_____	
_____	

3. Using the sample code 3 above write the complete program to find the maximum of three numbers and execute it for different set of values.

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date  /  /



**Set A: Apply all the three program development steps for the following examples.**

- Write a program to accept an integer and check if it is even or odd.
- Write a program to accept three numbers and check whether the first is between the other two numbers. Ex: Input 20 10 30. Output: 20 is between 10 and 30
- Accept a character as input and check whether the character is a digit. (Check if it is in the range '0' to '9' both inclusive)
- Write a program to accept a number and check if it is divisible by 5 and 7.
- Write a program, which accepts annual basic salary of an employee and calculates and displays the Income tax as per the following rules.
 

Basic: < 1,50,000	Tax = 0
1,50,000 to 3,00,000	Tax = 20%
> 3,00,000	Tax = 30%
- Accept a lowercase character from the user and check whether the character is a vowel or consonant. (Hint: a,e,i,o,u are vowels)

Signature of the instructor

Date  /  /



**Set B: Apply all the three program development steps for the following examples.**

1. Write a program to check whether given character is a digit or a character in lowercase or uppercase alphabet. (Hint ASCII value of digit is between 48 to 58 and Lowercase characters have ASCII values in the range of 97 to 122, uppercase is between 65 and 90)
2. Accept the time as hour, minute and seconds and check whether the time is valid. (Hint:  $0 \leq \text{hour} < 24$ ,  $0 \leq \text{minute} < 60$ ,  $0 \leq \text{second} < 60$ )
3. Accept any year as input through the keyboard. Write a program to check whether the year is a leap year or not. (Hint leap year is divisible by 4 and not by 100 or divisible by 400)
4. Accept three sides of triangle as input, and print whether the triangle is valid or not. (Hint: The triangle is valid if the sum of each of the two sides is greater than the third side).
5. Accept the x and y coordinate of a point and find the quadrant in which the point lies.
6. Write a program to calculate the roots of a quadratic equation. Consider all possible cases.
7. Accept the cost price and selling price from the keyboard. Find out if the seller has made a profit or loss and display how much profit or loss has been made.

Signature of the instructor

Date

**Set C: Write programs to solve the following problems**

1. Write a program to accept marks for three subjects and find the total marks secured , average and also display the class obtained. (Class I – above \_\_%, class II – \_\_% to \_\_%, pass class – \_\_% to \_\_% and fail otherwise)
2. Write a program to accept quantity and rate for three items, compute the total sales amount, Also compute and print the discount as follows: (amount > \_\_\_\_ – 20% discount, amount between \_\_\_\_ to \_\_\_\_ -- 15% discount, amount between – \_\_\_\_ to \_\_\_\_ -- 8 % discount)
3. A library charges a fine for every book returned late. Accept the number of days the member is late, compute and print the fine as follows:(less than five days Rs \_\_\_\_ fine, for 6 to 10 days Rs. \_\_\_\_ fine and above 10 days Rs. \_\_\_\_ fine )

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

**Assignment Evaluation**

**Signature**

0: Not done

2: Late Complete

4: Complete

1: Incomplete

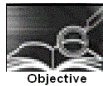
3: Needs improvement

5: Well Done

### Exercise 3

Start Date

/
/



**To demonstrate decision making statements (switch case)**



You should read following topics before starting this exercise

1. Different types of decision-making statements available in C.
2. Syntax for switch case statements.



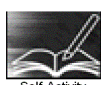
The control statement that allows us to make a decision from the number of choices is called a switch-case statement. It is a multi-way decision making statement.

#### 1. Usage of switch statement

Statement Syntax	Flowchart	Example
<pre> switch(expression) { case value1: block1;               break; case value2: block2;               break; . . . default : default block;               break; } </pre>	<pre> graph TD     Start([start]) --&gt; Case1{case 1}     Case1 -- True --&gt; Block1[Block 1]     Case1 -- False --&gt; Case2{case 2}     Case2 -- True --&gt; Block2[Block 2]     Case2 -- False --&gt; Case3{case 3}     Case3 -- True --&gt; Block3[Block 3]     Case3 -- False --&gt; Case4{case 4}     Case4 -- True --&gt; Block4[Block 4]     Case4 -- False --&gt; DefaultBlock[Default Block]     Block1 --&gt; Stop([stop])     Block2 --&gt; Stop     Block3 --&gt; Stop     Block4 --&gt; Stop     DefaultBlock --&gt; Stop </pre>	<pre> switch (color) { case 'r' : case 'R' : printf ("RED"); break; case 'g' : case 'G' : printf ("GREEN"); break; case 'b' : case 'B' : printf ("BLUE"); break; default : printf ("INVALID COLOR"); } </pre>

2. The switch statement is used in writing menu driven programs where a menu displays several options and the user gives the choice by typing a character or number. A Sample program to display the selected option from a menu is given below.

Step 1: Writing the Algorithm	Step 2: Draw the flowchart	Step 3: Writing Program
1. Start 2. Display the menu options 3. Read choice 4. Execute statement block depending on choice 5. Stop	<pre> graph TD     Start([start]) --&gt; Display[/Display Options/]     Display --&gt; Read[/Read choice/]     Read --&gt; Case1{case 1}     Case1 -- True --&gt; S1[Statement 1]     Case1 -- False --&gt; Case2{case 2}     Case2 -- True --&gt; S2[Statement 2]     Case2 -- False --&gt; Case3{case 3}     Case3 -- True --&gt; S3[Statement 3]     Case3 -- False --&gt; Default[Default statement]     S1 --&gt; Stop([stop])     S2 --&gt; Stop     S3 --&gt; Stop     Default --&gt; Stop           </pre>	<pre> /* Program using switch case and menu */  #include &lt;stdio.h&gt; main( ) { /* variable declarations */   int choice;   /* Displaying the Menu */   printf("\n 1. Option 1\n");   printf(" 2. Option 2\n");   printf(" 3. Option 3\n");   printf("Enter your choice");   scanf("%d",&amp;choice);   switch(choice)   {     case 1:       printf("Option 1 is selected");       break;     case 2:       printf("Option 2 is selected");       break;     case 3:       printf("Option 3 is selected");       break;     default:       printf("Invalid choice");   } }           </pre>



1. Write the program that accepts a char-type variable called color and displays appropriate message using the sample code 1 above. Execute the program for various character values and fill in the following table. Modify the program to include all rainbow colours

Input character	Output Message
V	
I	
B	
G	
R	

Signature of the instructor

Date  /  /



**Set A: Apply all the three program development steps for the following examples.**

1. Accept a single digit from the user and display it in words. For example, if digit entered is 9, display Nine.
2. Write a program, which accepts two integers and an operator as a character (+ - \* /), performs the corresponding operation and displays the result.
3. Accept two numbers in variables x and y from the user and perform the following operations

Options	Actions
1. Equality	Check if x is equal to y
2. Less Than	Check if x is less than y
3. Quotient and Remainder	Divide x by y and display the quotient and remainder
4. Range	Accept a number and check if it lies between x and y (both inclusive)
5. Swap	Interchange x and y

Signature of the instructor

Date

**Set B: Apply all the three program development steps for the following examples.**

1. Accept radius from the user and write a program having menu with the following options and corresponding actions

Options	Actions
1. Area of Circle	Compute area of circle and print
2. Circumference of Circle	Compute Circumference of circle and print
3. Volume of Sphere	Compute Volume of Sphere and print

2. Write a program having a menu with the following options and corresponding actions

Options	Actions
1. Area of square	Accept length ,Compute area of square and print
2. Area of Rectangle	Accept length and breadth, Compute area of rectangle and print
3. Area of triangle	Accept base and height , Compute area of triangle and print

Signature of the instructor

Date

**Set C: Write a program to solve the following problems**

1. Accept the three positive integers for date from the user (day, month and year) and check whether the date is valid or invalid. Run your program for the following dates and fill the table. (Hint: For valid date  $1 \leq \text{month} \leq 12, 1 \leq \text{day} \leq \text{no-of-days}$  where no-of-days is 30 in case of months 4, 6, 9 and 11. 31 in case of months 1, 3, 5, 7, 8, 10 and 12. In case of month 2 no-of-days is 28 or 29 depending on year is leap or not)

Date	Output
12-10-1984	
32-10-1920	
10-13-1984	
29-2-1984	
29-2-2003	
29-2-1900	

2. Write a program having menu that has three options - add, subtract or multiply two fractions. The two fractions and the options are taken as input and the result is displayed as output. Each fraction is read as two integers, numerator and denominator.

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

 **Assignment Evaluation****Signature**

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

## Exercise 4

Start Date

 /  / 


To demonstrate use of simple loops.



You should read following topics before starting this exercise

1. Different types of loop structures in C.
2. Syntax and usage of these statements.



We need to perform certain actions repeatedly for a fixed number of times or till some condition holds true. These repetitive operations are done using loop control statements. The types of loop structures supported in C are

1. while statement
2. do-while statement
3. for statement

Sr. No	Statement Syntax	Flowchart	Example
1.	<b>while statement</b>  <pre>while (condition) {     statement 1;     statement 2;     .     . }</pre>		<pre>/* accept a number*/ scanf("%d", &amp;n); /* if not a single digit */ while ( n &gt; 9) {     /* remove last digit     n = n /10; }</pre>
2.	<b>do-while statement</b>  <pre>do {     statement 1;     statement 2;     .     . } while (condition);</pre>		<pre>/*initialize sum*/ sum =0; do { /* Get a number */     printf( " give number");     scanf("%d",&amp;n);     /* add number to sum*/     sum=sum+n; } while ( n&gt;0); printf ( "sum is %d", sum);</pre>

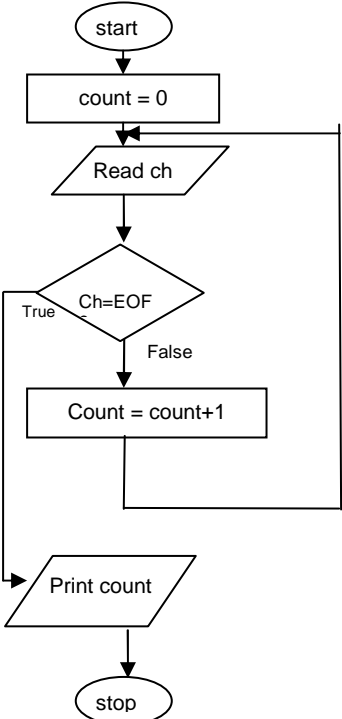
3.	<b>for statement</b>  for(expr1; expr2; expr3) { statement 1 . . } expr1 = initialization expression expr2 = loop condition expr3 = alteration expression which alters the loop variable	<pre> graph TD     Start(( )) --&gt; Expr1[expr1]     Expr1 --&gt; Test{Test expr2}     Test -- True --&gt; Body[Loop Body]     Body --&gt; Expr3[Expr3]     Expr3 --&gt; Test     Test -- False --&gt; Exit(( )) </pre>	/* display first 10 multiples of 2 */ for( i=1; i <= 10; i++) { printf ("2 X %d = %d\n", i, 2*i); } 
----	---	--	---

Note: Usually the for loop is used when the statements have to be executed for a fixed number of times. The while loop is used when the statements have to be executed as long as some condition is true and the do-while loop is used when we want to execute statements at least once (example: menu driven programs)

3. Sample program- to print sum of 1+2+3+.....n.

Step 1: Writing the Algorithm	Step 2: Draw the flowchart	Step 3: Writing Program
1. Start 2. Initialize sum to 0. 3. Accept n. 4. Compute sum=sum+n 5. Decrement n by 1 6. if n > 0 go to step 4 7. Display value of sum. 8. Stop	<pre> graph TD     Start([start]) --&gt; Sum0[Sum = 0]     Sum0 --&gt; ReadN[/Read n/]     ReadN --&gt; Compute[Compute Sum=sum+n]     Compute --&gt; Ngt0{n&gt;0}     Ngt0 -- True --&gt; Compute     Ngt0 -- False --&gt; Print[/Print value of sum/]     Print --&gt; Stop([stop]) </pre>	/* Program to calculate sum of numbers */  #include <stdio.h> main( ) { /* variable declarations */ int sum = 0, n; printf("enter the value of n : "); scanf("%d",&n); while (n>0) { sum = sum + n; n--; } printf("\n The sum of numbers is %d", sum); } 

4. Sample program- To read characters till EOF (Ctrl+Z) and count the total number of characters entered.

Step 1 : Writing the Algorithm	Step 2 : Draw the flowchart	Step 3 : Writing Program
<ol style="list-style-type: none"> <li>1. Start</li> <li>2. Initialize count to 0.</li> <li>3. Accept ch.</li> <li>4. If ch !=EOF     Count = count +1 Else     Go to step 6</li> <li>5. Go to step 3</li> <li>7. Display value of sum.</li> <li>8. Stop</li> </ol>	 <pre> graph TD     Start([start]) --&gt; Init[count = 0]     Init --&gt; Read[/Read ch/]     Read --&gt; Decision{Ch=EOF}     Decision -- True --&gt; Print[/Print count/]     Print --&gt; Stop([stop])     Decision -- False --&gt; Increment[Count = count+1]     Increment --&gt; Read     </pre>	<pre> /* Program to count number of characters */  #include &lt;stdio.h&gt; main( ) {     char ch;     int count=0;     while((ch=getchar())!=EOF)         count++;      printf("Total characters = %d", count); }     </pre>



1. Write a program that accepts a number and prints its first digit. Refer sample code 1 given above. Execute the program for different values.
2. Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the numbers read. Refer sample code 2 given above. Execute the program for different values.
3. Write a program to accept n and display its multiplication table. Refer to sample code 3 given above.
4. Type the sample program to print sum of first n numbers and execute the program for different values of n.
5. Write a program to accept characters till the user enters EOF and count number of times 'a' is entered. Refer to sample program 5 given above.

Signature of the instructor

Date / /





**Set A . Apply all the three program development steps for the following examples.**

1. Write a program to accept an integer n and display all even numbers upto n.
2. Accept two integers x and y and calculate the sum of all integers between x and y (both inclusive)
3. Write a program to accept two integers x and n and compute  $x^n$
4. Write a program to accept an integer and check if it is prime or not.
5. Write a program to accept an integer and count the number of digits in the number.
6. Write a program to accept an integer and reverse the number. Example: Input: 546, Output 645.
7. Write a program to accept a character, an integer n and display the next n characters.

Signature of the instructor

Date  /  /

**Set B. Apply all the three program development steps for the following examples.**

1. Write a program to display the first n Fibonacci numbers. (1 1 2 3 5 .....)
2. Write a program to accept real number x and integer n and calculate the sum of first n terms of the series  $x + 3x + 5x + 7x + \dots$
3. Write a program to accept real number x and integer n and calculate the sum of first n terms of the series  $\frac{1}{x} + \frac{2}{x^2} + \frac{3}{x^3} + \dots$
4. Write a program to accept characters till the user enters EOF and count number of alphabets and digits entered. Refer to sample program 5 given above.
5. Write a program, which accepts a number n and displays each digit in words. Example: 6702 Output = Six-Seven-Zero-Two. (Hint: Reverse the number and use a switch statement)

Signature of the instructor

Date  /  /

**Set C. Write C programs to solve the following problems**

1. Write a program to accept characters from the user till the user enters \* and count the number of characters, words and lines entered by the user. (Hint: Use a flag to count words. Consider delimiters like \n \t , ; . and space for counting words)
2. Write a program which accepts a number and checks if the number is a palindrome (Hint number = reverse of number)  
Example: number = 3472 Output: It is not a palindrome  
number = 262, Output : It is a palindrome
3. A train leaves station A at 4.00 a.m and travels at 80kmph. After every 30 minutes, it reaches a station where it halts for 10 minutes. It reaches its final destination B at 1.00 p.m. Display a table showing its arrival and departure time at every intermediate station. Also calculate the total distance between A and B.

4. A task takes  $4\frac{1}{2}$  hours to complete. Two workers, A and B start working on it and take turns alternately. A works for 25 minutes at a time and is paid Rs 50, B works for 75 minutes at a time and is paid Rs. 150. Display the total number of turns taken by A and B, calculate their total amounts and also the total cost of the task.

Signature of the instructor

Date  /  /

#### Assignment Evaluation

#### Signature

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

## Exercise 5

Start Date

/ /



To demonstrate use of nested loops



In the previous exercise, you used while, do-while and for loops. You should read following topics before starting this exercise

1. Different types of loop structures in C.
2. Syntax for these statements.
3. Usage of each loop structure



Nested loop means a loop that is contained within another loop. Nesting can be done upto any levels. However the inner loop has to be completely enclosed in the outer loop. No overlapping of loops is allowed.

Sr. No	Format	Sample Program
1.	<b>Nested for loop</b>  <pre>for(exp1; exp2 ; exp3) { .....   for(exp11; exp12 ; exp13)   { .....   }   ..... }</pre>	<pre>/* Program to display triangle of numbers*/ #include &lt;stdio.h&gt; void main( ) {   int n , line_number , number;   printf("How many lines: ");   scanf("%d",&amp;n);   for(line_number =1 ;line_number &lt;=n;   line_number++)   {     for(number = 1; number &lt;= line_number;     number++)       printf ("%dt", number);     printf ("\n");   } }</pre>
2.	<b>Nested while loop / do while loop</b>  <pre>while(condition1) { ..... while(condition2) { ..... } ..... }  do { .....   while(condition1)   { .....   } }</pre>	<pre>/* Program to calculate sum of digits till sum is a single digit number */ #include &lt;stdio.h&gt; void main( ) {   int n , sum;   printf("Give any number ");   scanf("%d",&amp;n);   do   {     sum =0;     printf("%d ---&gt;",n);     while ( n&gt;0)     { sum +=n%10;</pre>

..... } while (condition2);	n= n/10; } n=sum; } while( n >9); printf ( " %d" , n); }
--------------------------------	---

Note: It is possible to nest any loop within another. For example, we can have a for loop inside a while or do while or a while loop inside a for.



1. The Sample program 1 displays n lines of the following triangle. Type the program and execute it for different values of n.

```

1
1 2
1 2 3
1 2 3 4

```

2. Modify the sample program 1 to display n lines of the Floyd's triangle as follows (here n=4).

```

1
2 3
4 5 6
7 8 9 10

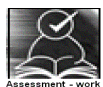
```

3. The sample program 2 computes the sum of digits of a number and the process is repeated till the number reduces to a single digit number. Type the program and execute it for different values of n and give the output

Input number	Output
6534	
67	
8	
567	

Signature of the instructor

Date

 /  / 


### Set A . Write C programs for the following problems.

1. Write a program to display all prime numbers between \_\_\_\_ and \_\_\_\_.

2. Write a program to display multiplication tables from \_\_\_\_ to \_\_\_\_ having n multiples each. The output should be displayed in a tabular format. For example, the multiplication tables of 2 to 9 having 10 multiples each is shown below.

2 × 1 = 2	3 × 1 = 3	.....9 × 1 = 9
2 × 2 = 4	3 × 2 = 6	.....9 × 2 = 18
.....	.....	.....
2 × 10 = 20	3 × 10 = 30	.....9 × 10 = 90

3. Modify the sample program 1 to display n lines as follows (here n=4).

```

A      B      C      D
E      F      G
H      I
J

```

Signature of the instructor

Date  /  /

**Set B. Write C programs for the following problems.**

1. Write a program to display all Armstrong numbers between 1 and 500. (An Armstrong number is a number such that the sum of cube of digits = number itself Ex.  $153 = 1*1*1 + 5*5*5 + 3*3*3$ )
2. Accept characters till the user enters EOF and count the number of lines entered. Also display the length of the longest line. (Hint: A line ends when the character is `\n`)
3. Display all perfect numbers below 500. [A perfect number is a number, such that the sum of its factors is equal to the number itself]. Example: 6 (1 + 2 + 3), 28 (1+2+4+7+14)

Signature of the instructor

Date  /  /

**Set C. Write C programs to solve the following problems**

1. A company has four branches, one in each zone: North, South, East and West. For each of these branches, it collects sales information once every quarter (four months) and calculates the average sales for each zone. Write a program that accepts sales details for each quarter in the four branches and calculate the average sales of each branch.
2. A polynomial in one variable is of the form  $a_0 + a_1x + a_2x^2 + \dots$ . For example,  $6 - 9x + 2x^5$ . Write a program to calculate the value of a polynomial. Accept the number of terms  $n$ , the value of  $x$ , and  $n+1$  coefficients.
3. The temperature of a city varies according to seasons. There are four seasons – spring, summer, Monsoon and winter. The temperature ranges are: Spring (15-25 degrees), Summer (25-40 degrees), Monsoon (20-35 degrees), Winter (5-20 degrees). Accept weekly temperatures (12 weeks per season) for each season, check if they are in the correct range and calculate the average temperature for each season.

Signature of the instructor

Date  /  /

**Assignment Evaluation**

**Signature**

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

## Exercise 6

Start Date

/ /



**To demonstrate menu driven programs and use of standard library functions**



You should read following topics before starting this exercise

1. Use of switch statement to create menus as in exercise 3
2. Use of while and do while loops as in exercise 4



A function is a named sub-module of a program, which performs a specific, well-defined task. It can accept information from the calling function in the form of arguments and return only 1 value. C provides a rich library of standard functions. We will explore some such function libraries and use these functions in our programs.

**ctype.h** : contains function prototypes for performing various operations on characters. Some commonly used functions are listed below.

Function Name	Purpose	Example
isalpha	Check whether a character is a alphabet	if (isalpha(ch))
isalnum	Check whether a character is alphanumeric	if (isalnum(ch))
isdigit	Check whether a character is a digit	if (isdigit(ch))
isspace	Check whether a character is a space	if (isspace(ch))
ispunct	Check whether a character is a punctuation symbol	if (ispunct(ch))
isupper	Check whether a character is uppercase alphabet	if (isupper(ch))
islower	Check whether a character is lowercase alphabet	if (islower(ch))
toupper	Converts a character to uppercase	ch = toupper(ch)
tolower	Converts a character to lowercase	ch = tolower(ch)

**math.h** : This contains function prototypes for performing various mathematical operations on numeric data. Some commonly used functions are listed below.

Function Name	Purpose	Example
cos	cosine	$a^2 + b^2 - 2ab \cos(\text{angle})$
exp(double x)	exponential function, computes $e^x$	exp(x)
log	natural logarithm	$c = \log(x)$
log10	base-10 logarithm	$y = \log_{10}(x)$
pow(x,y)	compute a value taken to an exponent, $x^y$	$y = 3^{\text{pow}(x, 10)}$
sin	sine	$z = \sin(x) / x$
sqrt	square root	$\text{delta} = \sqrt{b^2 - 4ac}$

Note: If you want to use any of the above functions you must include the library for example

```
#include <ctype.h>
```

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

In case of math library , it needs to be linked to your program. You have to compile the program as follows

```
$ cc filename -lm
```

A program that does multiple tasks, provides a menu from which user can choose the appropriate task to be performed. The menu should appear again when the task is completed so that the user can choose another task. This process continues till the user decides to quit. A menu driven program can be written using a combination of do-while loop containing a switch statement. One of the options provided in a menu driven program is to exit the program.

Statement Syntax	Flowchart	Example
<pre>do { display menu; accept choice; switch(choice) { case value1: block1;  break; case value2: block2;  break; . . . default : default block; } }while(choice != exit);</pre>	<pre>graph TD     Start([start]) --&gt; Display[Display menu]     Display --&gt; Accept[/Accept choice/]     Accept --&gt; Case1{case 1}     Case1 -- True --&gt; Block1[block1]     Case1 -- False --&gt; Case2{case 2}     Case2 -- True --&gt; Block2[block 2]     Case2 -- False --&gt; Default[default block]     Block1 --&gt; Exit{choice=exit?}     Block2 --&gt; Exit     Default --&gt; Exit     Exit -- True --&gt; Stop([stop])     Exit -- False --&gt; Accept</pre>	<pre>ch = getchar(); do { printf("\n 1: ISUPPER "); printf("\n 2: ISLOWER "); printf("\n 3: ISDIGIT "); printf("\n 4: EXIT");  printf("Enter your choice :"); scanf("%d", &amp;choice);  switch (choice) { case 1: if(isupper(ch)) printf("Uppercase"); break; case 2: if(islower(ch)) printf("Lowercase"); break; case 3: if(isdigit(ch)) printf("Digit"); break; } }while (choice!=4);</pre>



Self-Activity

1. Write a menu driven program to perform the following operations on a character type variable.

- i. Check if it is an alphabet
- ii. Check if it is a digit.
- iii. Check if it is lowercase.
- iv. Check if it is uppercase.
- v. Convert it to uppercase.
- vi. Convert it to lowercase.

Refer to the sample code given above and use standard functions from ctype.h



Assessment - Work

### Set A . Write C programs for the following problems

1. Write a program, which accepts a character from the user and checks if it is an alphabet, digit or punctuation symbol. If it is an alphabet, check if it is uppercase or lowercase and then change the case.

2. Write a menu driven program to perform the following operations till the user selects Exit. Accept appropriate data for each option. Use standard library functions from math.h  
i. Sine                      ii. Cosine                      iii. log                      iv.  $e^x$                       v. Square Root                      vi. Exit

3. Accept two complex numbers from the user (real part, imaginary part). Write a menu driven program to perform the following operations till the user selects Exit.  
i. ADD                      ii. SUBTRACT                      iii. MULTIPLY                      iv. EXIT

Signature of the instructor

Date  /  /

### Set B . Write C programs for the following problems

1. Accept x and y coordinates of two points and write a menu driven program to perform the following operations till the user selects Exit.

- Distance between points.
- Slope of line between the points.
- Check whether they lie in the same quadrant.
- EXIT

(Hint: Use formula  $m = (y_2 - y_1) / (x_2 - x_1)$  to calculate slope of line.)

2. Write a simple menu driven program for a shop, which sells the following items:  
The user selects items using a menu. For every item selected, ask the quantity. If the quantity of any item is more than 10, give a discount of \_\_\_\_%. When the user selects Exit, display the total amount.

Item	Price

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date  /  /

### Set C . Write C programs for the following problems

1. Write a program to calculate the total price for a picnic lunch that a user is purchasing for her group of friends. She is first asked to enter a budget for the lunch. She has the option of buying apples, cake, and bread. Set the price per kg of apples, price per cake, and price per loaf of bread in constant variables. Use a menu to ask the user what item and how much of each item she would like to purchase. Keep calculating the total of the items purchased. After purchase of an item, display the remaining amount. Exit the menu if the total has exceeded the budget. In addition, provide an option that allows the user to exit the purchasing loop at any time.

Signature of the instructor

Date  /  /

### Assignment Evaluation

### Signature

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>



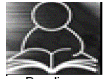
## Exercise 7

Start Date

/ /



**To demonstrate writing C programs in modular way (use of user defined functions)**



You should read following topics before starting this exercise

1. Declaring and Defining a function
2. Function call
3. Passing parameters to a function
4. Function returning a value



You have already used standard library functions. C allows to write and use user defined functions. Every program has a function named main. In main you can call some functions which in turn can call other functions.

The following table gives the syntax required to write and use functions

Sr. No	Actions involving functions	Syntax	Example
1.	Function declaration	returntype function(type arg1, type arg2 ... );	void display(); int sum(int x, int y);
2.	Function definition	returntype function(type arg1, type arg2 ... ) { /* statements*/ }	float calcarea (float r) { float area = Pi *r*r ; return area; }
3.	Function call	function(arguments); variable = function(arguments);	display(); ans = calcarea(radius);

### 1. Sample code

The program given below calculates the area of a circle using a function and uses this function to calculate the area of a cylinder using another function.

```
/* Program to calculate area of circle and cylinder using function */
```

```
#include <stdio.h>
void main()
{
    float areacircle (float r);
    float areacylinder(float r, int h);
    float area, r;
    printf("\n Enter Radius: ");
    scanf("%f",&r);

    area=areacircle(r);
    printf("\n Area of circle =%6.2f", area);

    printf("\n Enter Height: ");
```

```

scanf("%d",&h);
area=areacylinder(r,h);
printf("\n Area of cylinder =%6.2f", area);
}

float areacircle (float r)
{
    const float pi=3.142;
    return(pi * r*r );
}
float areacylinder (float r, int h)
{
    return 2*areacircle(r)*h;
}

```

## 2. Sample code

The function `isspace` returns 1 if its character parameter is a space, tab or newline character. The program accepts characters till the user enters EOF and counts the number of white spaces.

```

/* Program to count whitespaces using function */

#include <stdio.h>
void main()
{
    int isspace (char ch);
    char ch;
    int count=0;

    printf("\n Enter the characters. Type CTRL +Z to terminate: ");
    while((ch=getchar())!=EOF)
        if(isspace(ch))
            count++;
    printf("\n The total number of white spaces =%d", count);
}
int isspace (char ch)
{
    switch(ch)
    {
        case ' ':
        case '\t':
        case '\n': return 1;
        default : return 0;
    }
}

```



Self-Activity

1. Type the program given in sample code 1 above and execute the program. Comment function declarations and note down the type of error and the error messages received. Add another function to calculate the volume of sphere and display it.
2. Type the program given in sample code 2 above and execute the program. Comment function declaration and note down the type of error and the error messages received. Modify the function such that it returns 1 if the character is a vowel. Also count the total number of vowels entered.

**Set A . Write C programs for the following problems**

1. Write a function isEven, which accepts an integer as parameter and returns 1 if the number is even, and 0 otherwise. Use this function in main to accept n numbers and ckeck if they are even or odd.

2. Write a function, which accepts a character and integer n as parameter and displays the next n characters.

3. Write a function, which accepts a character and integer n as parameter and displays the next n characters.

Signature of the instructor

Date

**Set B . Write C programs for the following problems**

1. Write a function isPrime, which accepts an integer as parameter and returns 1 if the number is prime and 0 otherwise. Use this function in main to display the first 10 prime numbers.

2. Write a function that accepts a character as parameter and returns 1 if it is an alphabet, 2 if it is a digit and 3 if it is a special symbol. In main, accept characters till the user enters EOF and use the function to count the total number of alphabets, digits and special symbols entered.

3. Write a function power, which calculates  $x^y$ . Write another function, which calculates n! Using for loop. Use these functions to calculate the sum of first n terms of the Taylor series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \dots$$

Signature of the instructor

Date

**Set C . Write C programs for the following problems**

1. Write a menu driven program to perform the following operations using the Taylor series. Accept suitable data for each option. Write separate functions for each option.

i.  $e^x$ 

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \quad \text{for all } x$$

ii.  $\sin(x)$ 

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \quad \text{for all } x$$

iii.  $\cos(x)$ 

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots \quad \text{for all } x$$

Define separate functions to calculate  $x^y$  and n! and use them in each function.

Signature of the instructor

Date

**Assignment Evaluation****Signature**0: Not done 2: Late Complete 4: Complete 1: Incomplete 3: Needs improvement 5: Well Done

## Exercise 8

Start Date

/
/



### To demonstrate Recursion.



You should read the following topics before starting this exercise

1. Recursive definition
2. Declaring and defining a function
3. How to call a function
4. How to pass parameters to a function



Recursion is a process by which a function calls itself either directly or indirectly. The points to be remembered while writing recursive functions

- i. Each time the function is called recursively it must be closer to the solution.
- ii. There must be some terminating condition, which will stop recursion.
- iii. Usually the function contains an if –else branching statement where one branch makes recursive call while other branch has non-recursive terminating condition

Expressions having recursive definitions can be easily converted into recursive functions

Sr. No	Recursive definition	Recursive Function	Sample program
1.	<p>The recursive definition for factorial is given below:</p> $n! = 1 \quad \text{if } n = 0 \text{ or } 1$ $= n * (n-1)! \text{ if } n > 1$	<pre>long int factorial (int n) {     If (n==0)   (n==1))         /* terminating condition */         return(1);     else         return(n* factorial(n-1));         /* recursive call */ }</pre>	<pre>#include &lt;stdio.h&gt; main() {     int num;     /* function declaration */     long int factorial(int n);     printf("\n enter the number:");     scanf("%d",&amp;num);     printf("\n The factorial of %d is %ld",num,factorial(num)); } /* function code*/</pre>
2.	<p>The recursive definition for nCr ( no of combinations of r objects out of n objects) is as follows</p> $nCn = 1$ $nC0 = 1$ $nCr = n-1Cr + nCr-1$	<pre>long int nCr( int n, int r) { if(n==r    r==0)     /* terminating condition */     return(1);     else         return ( nCr(n-1,r) + nCr(n, r-1));         /* recursive call */ }</pre>	<pre>#include &lt;stdio.h&gt; /* function code*/ main() {     int n, r;     printf("\n enter the total number of objects:");     scanf("%d",&amp;n);     printf("\n enter the number of objects to be selected");     scanf("%d",&amp;r);     printf("\n The value %dC%d is %ld",n, r, nCr(n,r)); }</pre>



Self-Activity

1. Write the sample program 1 given above and execute the program. Modify the program to define a global integer variable count and increment it in factorial function. Add a printf statement in main function for variable count. Execute the program for different values and fill in the following table.

Sr. No.	num	factorial	Count
1.	0		
2	1		
3	5		
4	___		
5	___		

2. Write the sample program 2 given above and execute the program for different values of n and r. Modify the program to define a global integer variable count and increment it in nCr function. Add a print statement in main function for variable count. Execute the program for different values and fill in the following table

Sr. No.	n	r	nCr	Count
1.	5	0		
2	5	5		
3	5	2		
4	5	___		
5	___	___		

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date  /  /



Assessment - work

### Set A . Write C programs for the following problems

1. Write a recursive C function to calculate the sum of digits of a number. Use this function in main to accept a number and print sum of its digits.

2. Write a recursive C function to calculate the GCD of two numbers. Use this function in main. The GCD is calculated as :

$$\begin{aligned} \text{gcd}(a,b) &= a && \text{if } b = 0 \\ &= \text{gcd}(b, a \bmod b) && \text{otherwise} \end{aligned}$$

3. Write a recursive function for the following recursive definition. Use this function in main to display the first 10 numbers of the following series

$$\begin{aligned} a_n &= 3 && \text{if } n = 1 \text{ or } 2 \\ &= 2 * a_{n-1} + 3 * a_{n-2} && \text{if } n > 2 \end{aligned}$$

4. Write a recursive C function to calculate  $x^y$ . (Do not use standard library function)

Signature of the instructor

Date  /  /

### Set B . Write C programs for the following problems

1. Write a recursive function to calculate the nth Fibonacci number. Use this function in main to display the first n Fibonacci numbers. The recursive definition of nth Fibonacci number is as follows:

$$\begin{aligned} \text{fib}(n) &= 1 && \text{if } n = 1 \text{ or } 2 \\ &= \text{fib}(n-2) + \text{fib}(n-1) && \text{if } n > 2 \end{aligned}$$

2. Write a recursive function to calculate the sum of digits of a number till you get a single digit number. Example: 961 -> 16 -> 5. (Note: Do not use a loop)

3. Write a recursive C function to print the digits of a number in reverse order. Use this function in main to accept a number and print the digits in reverse order separated by tab.

Example 3456

6      5      4      3

(Hint: Recursiveprint(n) = print n if n is single digit number

= print n % 10 + tab + Recursiveprint( n/10)

Signature of the instructor

Date

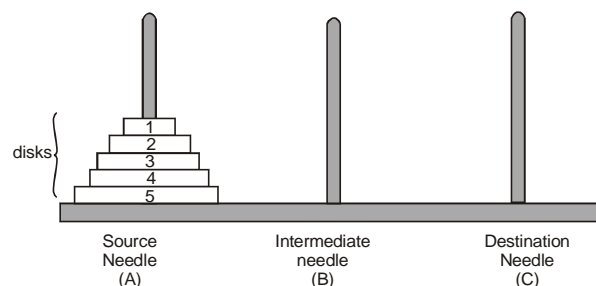
 /  / 

### Set C . Write C programs for the following problems

1. The "Towers of Hanoi" problem: The objective is to move a set of disks arranged in increasing sizes from top to bottom from the source pole to a destination pole such that they are in the same order as before using only one intermediate pole subject to the condition that

- Only one disk can be moved at a time
- A bigger disk cannot be placed on a smaller disk.

Write a recursive function which displays all the steps to move n disks from A to C.



Signature of the instructor

Date

 /  / 

### Assignment Evaluation

### Signature

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

## Exercise 9

Start Date

/ /



To demonstrate use of 1-D arrays and functions.



You should read the following topics before starting this exercise

1. What are arrays and how to declare an array?
2. How to enter data in to array and access the elements of an array.
3. How to initialize an array and how to check the bounds of an array?
4. How to pass an array to a function



An array is a collection of data items of the same data type referred to by a common name. Each element of the array is accessed by an index or subscript. Hence, it is also called a subscripted variable.

Actions involving arrays	syntax	Example
Declaration of array	<i>data-type array_name[size];</i>	int temperature[10]; float pressure[20];
Initialization of array	<i>data-type array_name[]={element1, element2, ....., element n};</i>  <i>data-type array_name[size]={element-1, element-2, ....., element-size};</i>  You cannot give more number of initial values than the array size. If you specify less values, the remaining will be initialized to 0.	int marks[]={45,57,87,20,90}; marks[3] refers to the fourth element which equals 20  int count[3]={4,2,9}; count[2] is the last element 9 while 4 is count[0]
Accessing elements of an array	The array index begins from <b>0 (zero)</b> To access an array element, we need to refer to it as array_name[index].	Value = marks[3]; This refers to the 4 <sup>th</sup> element in the array
Entering data into an array.		for (i=0; i<=9; i++) scanf("%d", &marks[i]);
Printing the data from an array		for(i=0; i<=9; i++) printf("%d", marks[i]);
Arrays and function	We can pass an array to a function using two methods. 1. Pass the array element by element 2. Pass the entire array to the function	/* Passing the whole array*/ void modify(int a[5]) { int i; for(i=0; i<5 ; i++) a[i] = i; }

Sample program to find the largest element of an array

```
/* Program to find largest number from array */

#include<stdio.h>
int main()
{
    int arr[20]; int n;
    void accept(int a[20], int n);
    void display(int a[20], int n);
    int maximum(int a[20], int n);

    printf("How many numbers :");
    scanf("%d", &n);
    accept(arr,n);
    display(arr,n);
    printf("The maximum is :%d", maximum(arr,n));
}

void accept(int a[20], int n)
{
    int i;
    for(i=0; i<n ; i++)
        scanf("%d", &a[i]);
}

void display(int a[20], int n)
{
    int i;
    for(i=0; i<n ; i++)
        printf("%d\t", a[i]);
}

int maximum(int a[20], int n)
{
    int i, max = a[0];

    for(i=1; i<n ; i++)
        if(a[i] > max)
            max = a[i];
    return max;
}
```



1. Write a program to accept n numbers in an array and display the largest and smallest number. Using these values, calculate the range of elements in the array. Refer to the sample code given above and make appropriate modifications.

2. Write a program to accept n numbers in an array and calculate the average. Refer to the sample code given above and make appropriate modifications.

Signature of the instructor

Date





**Set A. Write programs to solve the following problems**

1. Write a program to accept n numbers in the range of 1 to 25 and count the frequency of occurrence of each number.
2. Write a function for Linear Search, which accepts an array of n elements and a key as parameters and returns the position of key in the array and -1 if the key is not found. Accept n numbers from the user, store them in an array. Accept the key to be searched and search it using this function. Display appropriate messages.
3. Write a function, which accepts an integer array and an integer as parameters and counts the occurrences of the number in the array.
4. Write a program to accept n numbers and store all prime numbers in an array called prime. Display this array.

Signature of the instructor

Date  /  /

**Set B. Write programs to solve the following problems**

1. Write a program to accept n numbers from the user and store them in an array such that the elements are in the sorted order. Display the array. Write separate functions to accept and display the array. (Hint: Insert every number in its correct position in the array)
2. Write a function to sort an array of n integers using Bubble sort method. Accept n numbers from the user, store them in an array and sort them using this function. Display the sorted array.
3. Write a program to accept a decimal number and convert it to binary, octal and hexadecimal. Write separate functions.
4. Write a program to find the union and intersection of the two sets of integers (store it in two arrays).
5. Write a program to remove all duplicate elements from an array.

Signature of the instructor

Date  /  /

**Set C. Write programs to solve the following problems**

1. Write a program to merge two sorted arrays into a third array such that the third array is also in the sorted order.

a1	10	25	90						
a2	9	16	22	26	10				
					0				
a3	9	10	16	22	25	26	90	100	

2. Write a program to accept characters from the user till the user enters EOF and calculate the frequency count of every alphabet. Display the alphabets and their count.

Input: THIS IS A SAMPLE INPUT

Output: Character Count

T	2
H	1
I	3

.....

3. Write a recursive function for Binary Search, which accepts an array of n elements and a key as parameters and returns the position of key in the array and -1 if the key is not found. Accept n numbers from the user, store them in an array and sort the array. Accept the key to be searched and search it using this function. Display appropriate messages

Signature of the instructor

Date  /  /

**Assignment Evaluation**

**Signature**

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

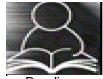
## Exercise 10

Start Date

/ /



To demonstrate use of 2-D arrays and functions.



You should read the following topics before starting this exercise

1. How to declare and initialize two-dimensional array
2. Accessing elements
3. Usage of two dimensional arrays



Actions involving 2-D arrays	syntax	Example
Declaration of 2-D array	<i>data-type array_name[size][size];</i>	int mat[10][10]; float sales[4][10];
Initialization of 2-D array	<i>data-type array_name[rows][cols]={  {elements of row 0},{ elements of row 1},.....}; data-type array_name[][cols]={element1,  element2, ....., element size};</i>	int num[][2] = {12, 34, 23, 45, 56, 45}; int num[3][2] = { {1,2}, {3,4}, {5,6}}; int num[3][2] = { 1,2,3,4, 5,6};
Accessing elements of 2-D array	Accessing elements of an two-dimensional array - in general, the array element is referred as array_name[index1][index2] where <i>index1</i> is the row location of and <i>index2</i> is the column location of an element in the array.	int m[3][2]; m is declared as a two dimensional array (matrix) having 3 rows (numbered 0 to 2) and 2 columns (numbered 0 to 1). The first element is m[0][0] and the last is m[2][1]. value = m[1][1];
Entering data into a 2-D array.		int mat[4][3]; for (i=0; i<4; i++) /* outer loop for rows */ for (j=0; j<3; j++) /* inner loop for columns */ scanf("%d", &mat[i][j]);
Printing the data from a 2-D array		for (i=0; i<4; i++) /* outer loop for rows */ { for (j=0; j<3; j++) /* inner loop for columns */ printf("%d\t", mat[i][j]); printf("\n"); }

Sample program to accept, display and print the sum of elements of each row of a matrix.

```
/* Program to calculate sum of rows of a matrix*/

#include<stdio.h>
int main()
{
    int mat[10][10], m, n;
    void display(int a[10][10], int m, int n);
    void accept(int a[10][10], int m, int n);
    void sumofrows(int a[10][10], int m, int n);

    printf("How many rows and columns? ");
    scanf("%d%d",&m, &n);

    printf("Enter the matrix elements :");
    accept(mat, m, n);
    printf("\n The matrix is :\n");
    display(mat, m, n);
    sumofrows(mat,m,n);
}

void accept(int a[10][10], int m, int n)
{
    int i,j;
    for (i=0; i<m; i++) /* outer loop for rows */
        for (j=0;j<n; j++) /* inner loop for columns */
            scanf("%d", &a[i][j]);
}

void display(int a[10][10], int m, int n)
{
    int i,j;
    printf("\nThe elements of %d by %d matrix are\n", m, n);
    for (i=0; i<m; i++) /* outer loop for rows */
    {
        for (j=0;j<n; j++) /* inner loop for columns */
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}

void somofrows(int a[10][10], int m, int n)
{
    int i,j, sum;
    for (i=0; i<m; i++) /* outer loop for rows */
    { sum=0
        for (j=0;j<n; j++) /* inner loop for columns */
            sum= sum+a[i][j];
        printf("Sum of elements of row %d = %d", i, sum);
    }
}
```



1. Write a program to accept, display and print the sum of elements of each row and sum of elements of each column of a matrix. Refer to sample code given above.

Signature of the instructor

Date



**Set A . Write C programs for the following problems.**

1. Write a program to accept a matrix A of size  $m \times n$  and store its transpose in matrix B. Display matrix B. Write separate functions.
2. Write a program to add and multiply two matrices. Write separate functions to accept, display, add and multiply the matrices. Perform necessary checks before adding and multiplying the matrices.

Signature of the instructor

Date  /  /

**Set B . Write C programs for the following problems.**

1. Write a menu driven program to perform the following operations on a square matrix. Write separate functions for each option.
  - i) Check if the matrix is symmetric.
  - ii) Display the trace of the matrix (sum of diagonal elements).
  - iii) Check if the matrix is an upper triangular matrix.

2. Write a menu driven program to perform the following operations on a square matrix. Write separate functions for each option.

- i) Check if the matrix is a lower triangular matrix.
- ii) Check if it is an identity matrix.

3. Write a program to accept an  $m \times n$  matrix and display an  $m+1 \times n+1$  matrix such that the  $m+1^{\text{th}}$  row contains the sum of all elements of corresponding row and the  $n+1^{\text{th}}$  column contains the sum of elements of the corresponding column.

Example:

A			B			
1	2	3	1	2	3	6
4	5	6	4	5	6	15
7	8	9	7	8	9	24
			12	15	18	45

Signature of the instructor

Date  /  /

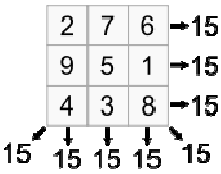
**Set C. Write programs to solve the following problems**

1. Pascal's triangle is a geometric arrangement of the binomial coefficients in a triangle. It is named after Blaise Pascal. Write a program to display n lines of the triangle.

```

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
```

2. A magic square of order  $n$  is an arrangement of  $n^2$  numbers, in a square, such that the  $n$  numbers in all rows, all columns, and both diagonals sum to the same constant. A normal magic square contains the integers from 1 to  $n^2$ . The magic constant of a magic square depends on  $n$  and is  $M(n) = (n^3+n)/2$ . For  $n=3,4,5$ , the constants are 15, 34, 65 resp. Write a program to generate and display a magic square of order  $n$ .



Signature of the instructor

Date  /  /

**Assignment Evaluation**

**Signature**

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>

**Exercise 11****Start Date**

/ /

**To demonstrate use of pointers in C.**

You should read the following topics before starting this exercise

1. What is a pointer?
2. How to declare and initialize pointers.
3. '\*' and '&' operators.
5. Pointer to a pointer.
6. Relationship between array and pointer.
7. Pointer to array and Array of pointers.
8. Dynamic memory allocation (malloc, calloc, realloc, free).



A Pointer is a variable that stores the memory address of another variable

<b>Actions involving Pointers</b>	<b>syntax</b>	<b>Example</b>
Declaration of pointers	<code>data_type * pointer_name</code>	<code>int *p1,*p2; float *temp1;</code>
Initialization of pointers	<code>pointer =&amp;variable p1=&amp;n;</code>	<code>int a, *p= &amp;a;</code>
Pointer Arithmetic	The C language allow arithmetic operations to be performed on pointers: Increment, Decrement, Addition, Subtraction When a pointer is incremented ( or decremented) by 1, it increments by sizeof(datatype). For example, an integer pointer increments by sizeof(int).	
Pointers and Functions	We can pass the address of a variable to a function. The function can accept this address in a pointer and use the pointer to access the variable's value.	
Arrays And Pointers	An array name is a pointer to the first element in the array. It holds the base address of the array. Every array expression is converted to pointer expression as follows: <code>a[i]</code> is same as <code>*(a+i)</code>	<code>int n; *n , *(n + 0 )</code> represents 0 <sup>th</sup> element <code>n[ j ] , *(n+ j ) , * ( j + n ) , j [ n ] :</code> represent the value of the j <sup>th</sup> element of array n
Pointer To Pointer		<code>int a; int * p; int **q; p = &amp;a;</code>

To allocate memory dynamically	<p>The functions used are : malloc, calloc, realloc</p> <p>ptr = ( cast-type * ) malloc ( byte-size ) ;</p> <p>Allocates a block of contiguous bytes. If the space in heap is not sufficient to satisfy request, allocation fails, returns NULL.</p> <p>ptr1 = ( cast-type * ) calloc ( byte-size);</p> <p>Similar to malloc, but initializes the memory block allocated to 0.</p> <p>ptr = realloc ( ptr, new size );</p> <p>To increase / decrease memory size.</p>	<pre>q = *p ; int * p,*p1; p = (int *) malloc(10 * sizeof(int)); p1 = (int *) calloc(10, sizeof(int)); p1=realloc(p1,20*sizeof(int));</pre>
--------------------------------	---	---

### 1. Sample program

```
/* Program to swap two numbers*/
main()
{
    int a = 10, b = 20;
    void swap1( int x, int y);
    void swap2( int *ptr1, int *ptr2);

    printf("\nBefore swapping : a=%d, b=%d", a,b);
    swap1(a, b);
    printf("\nAfter swapping by swap1 : a=%d, b=%d", a,b);
    swap2(&a, &b);
    printf("\nAfter swapping by swap2 : a=%d, b=%d", a,b);
}

void swap1( int x, int y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}

void swap2( int *ptr1, int *ptr2)
{
    int temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}
```



## 2. Sample program

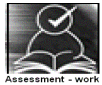
```
/* Program to allocate memory for n integers dynamically*/
#include <stdlib.h>
void main()
{
    int *p, n,i;
    printf("How many elements :");
    scanf("%d",&n);

    p = (int *)malloc(n*sizeof(int));
    /* Accepting data */
    for(i=0; i<n;i++)
        scanf("%d",p+i);

    /* Displaying data */
    for(i=0; i<n;i++)
        printf("%d\t",*(p+i));
}
```



1. Type the sample program 1 given above, execute it and write the output.
2. Sample program 2 allocates memory dynamically for n integers and accepts and displays the values. Type the sample program 2 given above, execute it. Modify the program to allocate memory such that the allocated memory is initialized to 0.



### Set A . Write C programs for the following problems.

1. Write a function which takes hours, minutes and seconds as parameters and an integer s and increments the time by s seconds. Accept time and seconds in main and Display the new time in main using the above function.
2. Write a program to display the elements of an array containing n integers in the reverse order using a pointer to the array.
3. Write a program to allocate memory dynamically for n integers such that the memory is initialized to 0. Accept the data from the user and find the range of the data elements.

Signature of the instructor

Date

### Set B . Write C programs for the following problems.

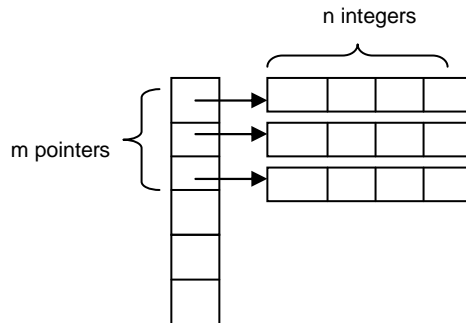
1. Accept n integers in array A. Pass this array and two counter variables to a function which will set the first counter to the total number of even values in the array and the other to the total number of odd values. Display these counts in main. (Hint: Pass the addresses of the counters to the function)
2. Write a function which accepts a number and three flags as parameters. If the number is even, set the first flag to 1. If the number is prime, set the second flag to 1. If the number is divisible by 3 or 7, set the third flag to 1. In main, accept an integer and use this function to check if it is even, prime and divisible by 3 or 7. (Hint : pass the addresses of flags to the function)

Signature of the instructor

Date

**Set C. Write programs to solve the following problems**

1. Accept the number of rows (m) and columns (n) for a matrix and dynamically allocate memory for the matrix. Accept and display the matrix using pointers. Hint: Use an array of pointers.



2. There are 5 students numbered 1 to 5. Each student appears for different number of subjects in an exam. Accept the number of subjects for each student and then accept the marks for each subject. For each student, calculate the percentage and display. (Hint: Use array of 5 pointers and use dynamic memory allocation)

Signature of the instructor

Date  /  /

**Assignment Evaluation**

**Signature**

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

## Exercise 12

Start Date

/ /



**To demonstrate strings in C.**



You should read the following topics before starting this exercise

1. String literals
2. Declaration and definition of string variables
3. The NULL character
4. Accepting and displaying strings
5. String handling functions



A string is an array of characters terminated by a special character called NULL character(\0). Each character is stored in 1 byte as its ASCII code.

Actions Involving strings	Explanation	Example
Declaring Strings		<code>char message[80];</code>
Initializing Strings		<code>char message[] = { 'H', 'e', 'l', 'l', 'o', '\0' } ;</code> <code>char message [ ] = "Hello";</code>
Accepting Strings	scanf and gets can be used to accept strings	<code>char name[20], address[50];</code> <code>printf("\n Enter your name :);</code> <code>scanf("%s", name);</code> <code>printf("\n Enter your address :);</code> <code>gets(address);</code>
Displaying Strings	printf and puts can be used to display strings.	<code>printf("\n The name is %s:",</code> <code>name);</code> <code>printf("\n The address is :");</code> <code>puts(address);</code>
String functions	All string operations are performed using functions in "string.h". Some of the most commonly used functions are a. strlen – Returns the number of characters in the string (excluding \0) b. strcpy – Copies one string to another c. strcmp – Compares two strings. Returns 0 (equal), +ve (first string > second), -ve (first string < second ). It is case sensitive d. strcmpi – Same as strcmp but ignores case e. strcat – Concatenates the second string to the first.	<code>#include &lt;string.h&gt;</code> <code>main( )</code> <code>{</code> <code>char str1[50], str2[50],str3[100];</code> <code>printf("\n Give the first string:");</code> <code>gets(str1);</code> <code>printf("\n Give the second string</code> <code>string:");</code> <code>gets(str2);</code> <code>if (strlen(str1) == strlen(str2)</code> <code>{strcpy(str3, strrev(str1));</code> <code>strcat(str3,strupr(str2));</code> <code>puts(strupr(str3));</code> <code>}</code> <code>else</code> <code>puts(strlwr(str2);</code> <code>}</code>

	Returns the concatenated string. f. <code>strrev</code> – Reverses a string and returns the reversed string. g. <code>strupr</code> – Converts a string to uppercase. h. <code>strlwr</code> - Converts a string to lowercase	
--	--	--

### Sample program :

The following program demonstrates how to pass two strings to a user defined function and copy one string to other using pointers

```

void string_copy (char *t,char *s)
{
    while (*s !='\0')          /* while source string does not end */
    {
        *t=*s;
        s++;
        t++;
    }
    *t ='\0'; /*      terminate target string */
}

void main()
{
    char str1[20], str2[20];
    printf("Enter a string :");
    gets(str1);
    string_copy(str2, str1);
    printf("The copied string is :");
    puts(str2);
}

```



1. Write a program to accept two strings `str1` and `str2`. Compare them. If they are equal, display their length. If `str1 < str2`, concatenate `str1` and the reversed `str2` into `str3`. If `str1 > str2`, convert `str1` to uppercase and `str2` to lowercase and display. Refer sample code for string functions above.

2. Type the sample program above and execute it. Modify the program to copy the characters after reversing the case. (Hint: First check the case of the character and then reverse it)

Signature of the instructor

Date

 /  / 


### Set A . Write C programs for the following problems.

1. Write a menu driven program to perform the following operations on strings using standard library functions:

Length	Copy	Concatenation	Compare
Reverse	Uppercase	Lowercase	Check case

2. Write a program that will accept a string and character to search. The program will call a function, which will search for the occurrence position of the character in the

string and return its position. Function should return -1 if the character is not found in the string.

3. A palindrome is a string that reads the same-forward and reverse. *Example:* "madam" is a Palindrome. Write a function which accepts a string and returns 1 if the string is a palindrome and 0 otherwise. Use this function in main.

4. For the following standard functions, write corresponding user defined functions and write a menu driven program to use them. strcat, strcmp, strrev, strupr

5. Write a program which accepts a sentence from the user and alters it as follows: Every space is replaced by \*, case of all alphabets is reversed, digits are replaced by ?

Signature of the instructor

Date

**Set B . Write C programs for the following problems.**

1. Write a menu driven program which performs the following operations on strings. Write a separate function for each option. Use pointers

- i. Check if one string is a substring of another.
- ii. Count number of occurrences of a character in the string.
- iii. Replace all occurrences of a character by another.

2. Write a program in C to reverse each word in a sentence.

3. Write a function which displays a string in the reverse order. (Use recursion)

Signature of the instructor

Date

**Set C. Write programs to solve the following problems**

1. Write a program that accepts a sentence and returns the sentence with all the extra spaces trimmed off. (In a sentence, words need to be separated by only one space; if any two words are separated by more than one space, remove extra spaces)

2. Write a program that accepts a string and displays it in the shape of a kite. Example: "abcd" will be displayed as :

```
aa
abab
abcabc
abcdabcd
abcabc
abab
aa
```

3. Write a program that accepts a string and generates all its permutations. For example: ABC, ACB, BAC, BCA, CAB, CBA

4. Write a program to display a histogram of the frequencies of different characters in a sentence. Note: The histogram can be displayed as horizontal bars constructed using \* character. Example: this is a single string

t       \*       \*  
 h       \*  
 i       \*       \*       \*       \*  
 s       \*       \*       \*       \*  
 a       \*       \*  
 n       \*       \*  
 g       \*       \*  
 l       \*  
 e       \*  
 r       \*

Signature of the instructor

Date

**Assignment Evaluation**

**Signature**

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>

### Exercise 13

Start Date

/ /



#### To demonstrate array of Strings.



You should read the following topics before starting this exercise

1. How to declare and initialize strings.
2. String handling functions
3. How to create and access an array of strings.
4. Dynamic memory allocation



An array of strings is a two dimensional array of characters. It can be treated as a 1-D array such that each array element is a string.

Actions Involving array of strings	Explanation	Example
Declaring String array	char array[size1][size2];	char cities[4][10]
Initializing String array		char cities[4][10] = { "Pune", "Mumbai", "Delhi", "Chennai" };

#### Sample program-

The following program illustrates how to accept 'n' names , store them in an array of strings and search for a specific name.

```
/* Program to search for name from array */
#include <stdio.h>
void main( )
{
    char list[10][20]; /*list is an array of 10 strings */
    char name[20];
    int i,n;
    printf("\n How many names ?:");
    scanf("%d", &n);
    for (i=0;i<n; i++)
    {
        printf("\n Enter name %d,i);
        scanf("%s", list[i]);
    }
    printf("\n The names in the list are : \n");
    for (i=0; i<n; i++)
        printf("%s", list[i]);
    printf("\n Enter the name to be searched ");
    scanf("%s", name);
    for (i=0; i<n; i++)
        if(strcmp(list[i],name)==0)
        {
            printf("Match found at position %d", i);
            break;
        }
    if(i==n)
        printf("Name is not found in the list");
}
```



1. Type the above sample program and execute the same for different inputs.

Signature of the instructor

Date  /  /



### Set A . Write C programs for the following problems.

1. Write a program that accepts n words and outputs them in dictionary order.
2. Write a program that accepts n strings and displays the longest string.
3. Write a program that accepts a sentence and splits the sentence into words. Sort each word and reconstruct the sentence.

Input – this is a string                      Output – hist is a ginrst

Signature of the instructor

Date  /  /

### Set B . Write C programs for the following problems.

1. Write a function, which displays a given number in words.  
For Example:   129     One Hundred Twenty Nine  
                  2019    Two Thousand Nineteen
2. Define two constant arrays of strings, one containing country names (ex: India, France etc) and the other containing their capitals. (ex: Delhi, Paris etc). Note that country names and capital names have a one-one correspondence. Accept a country name from the user and display its capital. Example: Input: India , Output: Delhi.

Signature of the instructor

Date  /  /

### Set C. Write programs to solve the following problems

1. Create a mini dictionary of your own. Each entry in the dictionary contains three parts (word, its meaning, similar word). The entries are stored in the sorted order of words. Write a menu driven program, which performs the following operations.
  - i. Add a new word (Insert new word and its details in the correct position)
  - ii. Dictionary look-up
  - iii. Find similar word
  - iv. Delete word
  - v. Display All words starting with a specific alphabet (along with their meaning).

(Hint: Use 2-D array of strings having n rows and 3 columns)

Signature of the instructor

Date  /  /

### Assignment Evaluation

### Signature

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>



## Exercise 14

Start Date

/ /



**Assignment to demonstrate bitwise operators.**



You should read the following topics before starting this exercise

1. Bitwise operators and their usage ( &, |, ^, ~, <<, >>)



**1. Bitwise operators:** C provides 6 operators to perform operations on bits. These operators operate on integer and character but not the float and double. Ones complement operator (~) is unary while the others are binary.

Operator	Purpose	Example
~	One's complement	~a : Complements each bit of variable a
>>	Right shift	a=a>>1; Shifts bits of a one position to the right
<<	Left Shift	a=a<<n; Shifts bits of a n positions to the left
&	Bitwise AND	a = b&c; performs bitwise AND on b and c a = a&0xFF00; Masks the lower order 8 bits of a
	Bitwise OR	a = a b; performs bitwise OR on b and c
^	Bitwise XOR	x = x^y; y=x^y; x=x^y; Swaps x and y by performing bitwise XOR.

**Sample code:** The following function accepts an integer argument and displays it in binary format. It uses shift operator and AND masking.

```
void displaybits(unsigned int n)
{
    unsigned int mask = 32768;
    /*set MSB of mask to 1 */
    while (mask>0)
    {
        if((n&mask)==0)
            printf("0");
        else
            printf("1");
        mask = mask >>1; /* shift mask right */
    }
}
```



1. Write a program to accept n integers and display them in binary. Use the function given above.

Signature of the instructor

Date

/ /



### Set A . Write C programs for the following problems.

1. Write a program to accept 2 integers and perform bitwise AND, OR, XOR and Complement. Display the inputs and results in binary format. Use the function in the above exercise.
2. Write a program to swap two variables without using a temporary variable. (Hint: Use XOR)
3. Write a program which accepts two integers x and y and performs  $x \ll y$  and  $x \gg y$ . Display the result in binary.

Signature of the instructor

Date

### Set B . Write C programs for the following problems.

1. Write functions to calculate the size of an integer, character, long and short integer using bitwise operators. Store their declaration in file "myfunctions.h" and their definitions in file "myfunctions.c". Include these files in your program and use these functions to display the size of each.
2. Write a program to perform the following operations on an unsigned integer using bitwise operators and display the result in hexadecimal format.
  - i. Swap the \_\_\_\_ and \_\_\_\_ nibble ( 4 bits)
  - ii. Remove the lower order nibbles from the number.  
For example: Input: A3F1 Output 00A3
  - iii. Reverse the nibbles  
For example: Input: A3F1 Output 1F3A
3. Write a program which accepts an integer and checks whether it is a power of 2.

Signature of the instructor

Date

### Set C. Write programs to solve the following problems

1. Write a program to add, subtract, multiply and divide two integers using bitwise operators.
2. Packing and Unpacking Data: A date consists of three parts : day, month, year. To store this information, we would require 3 integers. However, day and month can take only limited values. Hence, we can store all three in a single integer variable by packing bits together. If we are using the dd-mm-yy format, the date will be stored in memory as an unsigned integer (16 bits) in the following format. Year (Bits 15-9), Month (bits 8 – 5), Day (Bits 4 - 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
y	y	y	y	y	y	y	m	m	m	m	d	d	d	d	d
hour							Month				day				

Accept day, month and year from the user and pack them into a single unsigned int. Unpack and display them in the binary format. (Hint: for packing, use:  $512 * \text{year} + 32 * \text{month} + \text{day}$ )

The output should be:

Enter the date, month and year –dd mm yy :

31      12      89

Packed date = 1011001110011111

Day = 31

0000000000011111

Month = 12  
0000000000001100  
Year = 89  
0000000001011001

3. Packing and Unpacking Data: Time consists of three parts : hours, minutes, seconds. To store this information, we would require 3 integers. However, all these three variable take only limited values. Hence, we can store all three in a single integer variable by packing bits together. Time being 0 to 23 hours, it will require maximum 5 bits, minutes being 0 to 59 will require 6 bits. The two together take up 11 bits. The remaining 5 bits cannot store seconds which are also in the range 0 to 59 hence we store double seconds which are in the range 0 to 29

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
h	h	h	h	h	m	m	m	m	m	m	ds	ds	ds	ds	ds
hour					Minutes					Double seconds					

Accept hour, minute and double seconds from the user and pack them into a single unsigned int. Unpack and display them in the binary format.

The output should be:

Enter the hour, minutes and double seconds –hh mm ss :

07      12      20

Packed date = 0011100110010100

Hour = 07

0000000000000111

Minutes = 12

0000000000001100

Double seconds = 20

0000000000010100

Signature of the instructor

Date  /  /

#### Assignment Evaluation

#### Signature

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

**Exercise 15****Start Date**

/ /

**Structures in C**

You should read the following topics before starting this exercise

1. Concept of structure
2. Declaring a structure
3. Accessing structure members
4. Array of structures
5. Pointer to a structure.
6. Passing structures to functions



A structure is a composition of variables possibly of different data types, grouped together under a single name. Each variable within the structure is called a 'member'.

Operations performed	Syntax / Description	Example
Declaring a structure	<pre>struct structure-name {     type member-1 ;     type member-2;     .     .     type member-n ; };</pre>	<pre>struct student {     char name[20];     int rollno;     int marks; };</pre>
Creating structure variables	<code>struct structurename variable;</code>	<code>struct student stud1;</code>
Accessing structure members	<code>variable.member</code>	<pre>stud1.name stud1.rollno stud1.marks</pre>
initializing a structure variable	the initialization values have to be given in {} and in order	<code>struct student stud1 = {"ABCD",10,95};</code>
Pointer to a structure	<code>struct structure-name * pointer-name;</code>	<pre>struct student *ptr; ptr = &amp;stud1;</pre>
Accessing members using Pointer	<code>pointer-name -&gt; member-name;</code>	<code>ptr-&gt;name; ptr-&gt;rollno;</code>
Array of structures	<code>struct structure-name array-name[size];</code>	<code>struct student stud[10];</code>
passing Structures to Functions	<code>return-type function-name ( struct structure-name variable);</code>	<code>void display(struct student s);</code>
pass an array of structures to a function	<code>return-type function-name ( struct structure-name array[size]);</code>	<code>void display(struct student stud[10]);</code>

### Sample Code:

```
/* Program for student structure */

#include<stdio.h>
struct student
{
    char name[20];
    int rollno;
    int marks[3];
    float perc;
};
void main( )
{
    int i, sum j;
    struct student s[10];
    printf("\n Enter the details of the 10 students \n");
    for (i=0;i<10;i++)
    {
        printf("\n Enter the name and roll number \n");
        scanf("%s%d",s[i].name, &s[i].rollno);
        printf("\n Enter marks for three subjects:");
        sum = 0 ;
        for { j=0;j<3;j++)
        {
            scanf("%d",&s[i].marks[j]);
            sum = sum + s[i].marks[j];
        }
        s[i].perc = (float)sum/3;
    }
    /* Display details of students */
    printf("\n\n Name \t Roll no\t Percentage");
    printf("\n===== \n");
    for(i=0;i<10;i++)
    {
        printf("\n%s\t%d\t%f",s[i].name,s[i].rollno,s[i].perc);
    }
}
```

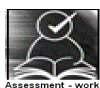


Self-Activity

1. The program in Sample code 1 demonstrates an array of structures of the type student. Type the above program and run it. Modify the program to display the details of the student having the highest percentage.

Signature of the instructor

Date

Assessment - work

### Set A . Write C programs for the following problems.

1. Create a structure student (roll number, name, marks of 3 subjects, percentage). Accept details of n students and write a menu driven program to perform the following operations. Write separate functions for the different options.

- i) Search
- ii) Modify
- iii) Display all student details
- iv) Display all student having percentage > \_\_\_\_\_
- v) Display student having maximum percentage

2. Create a structure employee (id, name, salary). Accept details of n employees and write a menu driven program to perform the following operations. Write separate functions for the different options.

- i) Search by name
- ii) Search by id
- iii) Display all
- iv) Display all employees having salary > \_\_\_\_\_
- v) Display employee having maximum salary

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date  /  /

### Set B . Write C programs for the following problems.

1. Create a structure having the following fields:

Structure name: \_\_\_\_\_

Fields: \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_

Accept details of n variables of the above structure and write a menu driven program to perform the following operations. Write separate functions for the different options.

- i) \_\_\_\_\_ ii) \_\_\_\_\_ iii) \_\_\_\_\_ iv) \_\_\_\_\_

2. Create a structure Fraction (numerator, denominator). Accept details of n fractions and write a menu driven program to perform the following operations. Write separate functions for the different options. Use dynamic memory allocation. Note: While accepting fractions, store the fractions in the reduced form.

- i) Display the largest fraction
- ii) Display the smallest fraction
- iii) Sort fractions
- iv) Display all

Signature of the instructor

Date  /  /

### Set C. Write programs to solve the following problems

1. Accept book details of 'n' books viz, book title, author, publisher and cost. Assign an accession numbers to each book in increasing order. (Use dynamic memory allocation). Write a menu driven program for the following options.

- i. Books of a specific author
- ii. Books by a specific publisher
- iii. All books having cost >= \_\_\_\_\_ .
- iv. Information about a particular book (accept the title)
- v. All books.

2. The government of a state wants to collect census information for each city and store the following information : city name, population of the city, literacy percentage. After collecting data for all cities in the state, the government wants to view the data according to :

- i. Literacy level
- ii. Population
- iii. Details of a specific city.

Write a C program using structures and dynamic memory allocation.

Signature of the instructor

Date  /  /

### Assignment Evaluation

### Signature

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

**Exercise 16****Start Date**

/ /

**Nested Structures and Unions**

You should read the following topics before starting this exercise

1. Creating and accessing structures
2. Array of structures
3. Dynamic memory allocation
4. Structure within a structure
5. Creating and accessing unions



Nested structures: The individual members of a structure can be other structures as well. This is called nesting of structures.

Operations performed	Syntax	Example
Creating a nested structure	<pre> struct structure1 {     ...     struct structure2     {         ...     } variable;     ... };  Method 2 struct structure2 {     ... };  struct structure1 {     ...     struct structure2     variable;     ... }; </pre>	<pre> struct student {     int rollno; char name[20];     struct date     {         int dd, mm, yy;     } bdate, admdate; };  struct date {     int dd, mm, yy; };  struct student {     int rollno; char name[20];     struct date bdate, admdate; }; </pre>
Accessing nested structure members	nested structure members can be accessed using the (.) operator repeatedly.	stud1.bdate.dd, stud1.bdate.mm
self referential structure	A structure containing a pointer to the same structure	<pre> struct node {     int info;     struct node *next; }; </pre>

		};
Unions	A union is a variable that contains multiple members of possibly different data types grouped together under a single name. However, only one of the members can be used at a time. They occupy the same memory area.	union u { char a; int b; };

### Sample Code 1:

Example: The following structure is for a library book with the following details : id, title, publisher, code ( 1 – Text book, 2 – Magazine, 3 – Reference book). If the code is 1, store no-of-copies. If code = 2, store the issue month name. If code = 3, store edition number. Also store the cost.

```
/* Program for demonstrating structure and union */

struct library_book
{
    int id;
    char title[80],publisher[20] ;
    int code;
    union u
    {
        int no_of_copies;
        char month[10];
        int edition;
    }info;
    int cost;
};

void main( )
{
    struct library_book book1;
    printf("\n Enter the details of the book \n");

    printf("\n Enter the id, title and publisher \n");
    scanf("%d%s%s",&book1.id, book1.title, book1.publisher);
    printf("\n Enter the code: 1-Text Book, 2-Magazine, 3-Reference");
    scanf("%d",book1.code);
    switch(book1.code)
    {
        case 1: printf("Enter the number of copies :");
                scanf("%d",&book1.info.no_of_copies);
                break;
        case 2:  printf("Enter the issue month name :");
                 scanf("%s",book1.info.month);
                 break;
        case 3:  printf("Enter the edition number:");
                 scanf("%d",&book1.info.edition);
                 break;
    }
    printf("Enter the cost :");
    scanf("%d",&book1.cost);

    /*      Display details of book */
    printf("\n id = %d", book1.id);
```



```

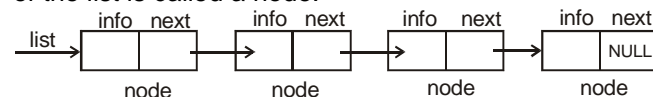
printf("\n Title = %s", book1.title);
printf("\n Publisher = %s", book1.publisher);
switch(book1.code)
{
    case 1:    printf("Copies = %d:", book1.info.no_of_copies);
               break;
    case 2:    printf("Issue month name = %s",book1.info.month);
               break;
    case 3:    printf("Edition number =%d:",book1.info.edition);
               break;
}
printf("\n Cost = %d", book1.cost);
}

```

### Sample Code 2:

A linked list is a collection of data elements which are linked to one another by using pointers i.e. the every node stores the address of the next node. The advantage of using a linked list over an array is that it is easy to insert and delete elements from the list.

To create a linked list, we have to use a self referential structure (See table above). Each element of the list is called a node.



To create a node, we have to allocate memory dynamically. The following program creates 5 nodes , stores the numbers 1...5 in them and displays the data.

```

/* Program to create a linked list of 5 nodes */

#include <stdio.h>
struct node
{
    int info;
    struct node *next;
};
struct node *list = NULL; /* list is a pointer to the linked list */

void createlist()
{
    struct node *temp, *p;
    int i;
    for(i=1;i<=5;i++)
    {
        p=(struct node *)malloc(sizeof(struct node)); /* create a node */
        p->info = i;
        p->next=NULL;
        if(list == NULL)
            list=temp=p; /* list points to the first node */
        else
        {
            temp->next=p; /* link new node to the last */
            temp=p;
        }
    }
}

void displaylist()
{
    struct node *temp;

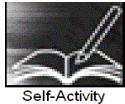
```

```

for(temp=list; temp!=NULL; temp=temp->next) /* use a temporary pointer */
    printf("%d \t", temp->info);
}

void main( )
{
    createlist();
    displaylist();
}

```

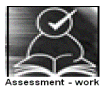


1. The sample code 1 given above demonstrates how we can create a variable of the above structure and accept and display details of 1 book. Type the program and execute it. Modify the program to accept and display details of n books.

2. The sample code 2 given above demonstrates how we can create a linked list and traverse the list. Type the program and execute it. Modify the displaylist function to display only the even numbers from the list.

Signature of the instructor

Date

 /  / 


### Set A . Write C programs for the following problems.

1. Modify the sample program 1 above to accept details for n books and write a menu driven program for the following:

- i) Display all text books
- ii) Search Text Book according to Title
- iii) Find the total cost of all books (Hint: Use no\_of\_copies).

2. Modify the sample program 1 to accept details for n books and write a menu driven program for the following:

- i) Display all magazines
- ii) Display magazine details for specific month.
- iii) Find the “costliest” magazine.

3. Modify the sample program 1 to accept details for n books and write a menu driven program for the following:

- i) Display all reference books
- ii) Find the total number of reference books
- iii) Display the edition of a specific reference book.

Signature of the instructor

Date

 /  /

**Set B. Write programs to solve the following problems**

1. Create a structure named \_\_\_\_\_ having the following fields:

Field name	Description

Write a menu driven program to perform the following operations :

i) \_\_\_\_\_ ii) \_\_\_\_\_ iii) \_\_\_\_\_ iv) \_\_\_\_\_ v) \_\_\_\_\_

2. Write a program to create a linked list of n nodes and accept data from the user for each node. Display the list. Accept a number from the user and search for the element in the list.

Signature of the instructor

Date

 /  / 
**Set C. Write programs to solve the following problems**

1. A shop sells electronic items. Each item has an id, company name, code (1-TV, 2-Mobile phones, 3-Camera) and cost. The following additional details are stored for each item.

- TV - size, type ( CRT-1 / LCD- 2 / Plasma-3)
- Mobile Phone - type ( GSM – 1 / CDMA – 2) , model number.
- Camera – resolution, model number.

The shop wants to maintain a list of all items and perform the following operations for each of the item types:

- i) Display all
- ii) Search for specific item
- iii) Sort according to cost

2. Write a program to create a linked list of n nodes and accept data from the user for each node. Write a menu driven program to perform the following operations:

- i) Display the list
- ii) Search for specific number
- iii) Display the element after \_\_\_\_\_
- iv) Find the maximum / minimum

Signature of the instructor

Date

 /  / 
**Assignment Evaluation****Signature**

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

## Exercise 17

Start Date

/ /



**Assignment to demonstrate command line arguments and preprocessor directives.**



You should read the following topics before starting this exercise

1. Passing arguments from the command line to main
2. Accessing command line arguments
3. File inclusion, macro substitution and conditional compilation directives.
4. Argumented and Nested macros



Preprocessor directives	They begin with a # which must be the first non-space character on the line. They do not end with a semicolon.	
Macro Substitution Directive	# define MACRO value	# define PI 3.142
Argumented macro	# define MACRO(argument) value	# define SQR(x) x*x #define LARGER(x,y) ((x)>(y)?(x):(y))
Nested macro	one macro using another	#define CUBE(x) (SQUARE(x)*(x))
File Inclusion directive	#include <filename> #include "filename"	#include <stdio.h>
Conditional Compilation directive	# if, # else, # elif, # endif #ifdef	#ifdef PI #undef PI #endif
Command Line Arguments	int argc - argument counter char *argv[]-argument vector	void main(int argc, char *argv[]) { printf("There are %d arguments in all", argc); for (i=0; i<argc; i++) printf("Argument %d =%s",i,argv[i]); }
To run a program using command line arguments	Compile the program using cc Execute the program using a.out followed by command line arguments	Example: a.out ABC 20 Here, ABC and 20 are the two command line arguments which are stored in the form of strings. To use 20 as an integer, use function atoi . Example: int num = atoi(argv[2]);

### Sample Code 1

```
/* Program for argumented macros */

#define INRANGE(m) ( m >= 1 && m<=12)
#define NEGATIVE(m) (m<0)
#define ISLOWER(c) (c>='a'&&c<='z')
#define ISUPPER(c) (c>='A'&&c<='Z')
#define ISALPHA(c) (ISUPPER(c)||ISLOWER(c))
#define ISDIGIT(c) (c>='0'&&c<='9')

void main()
{
    int m; char c;
    printf("Enter an integer corresponding to the month");
    scanf("%d",&m);
    if(NEGATIVE(m))
        printf("Enter a positive number");
    else
        if(INRANGE(m))
            printf("You Entered a valid month");

    printf("Enter a character :");
    c=getchar();
    if(ISAPLHA(c))
        printf("You entered an alphabet");
    else
        if(ISDIGIT(c))
            printf("You Entered a digit");
}
```

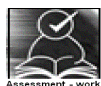


1. Write a program to display all command line arguments passed to main in the reverse order.  
Hint: See table above.

2. Sample code 1 above demonstrates the use of argumented and nested macros. Type the program and execute it.

Signature of the instructor

Date



### Set A . Write C programs for the following problems.

1. Write a program to accept three integers as command line arguments and find the minimum, maximum and average of the three. Display error message if invalid number of arguments are entered.

2. Write a program which accepts a string and two characters as command line arguments and replace all occurrences of the first character by the second.

3. Define a macro EQUALINT which compares two parameters x and y and gives 1 if equal and 0 otherwise. Use this macro to accept pairs of integers from the user. Calculate the sum of digits of both and continue till the user enters a pair whose sum of digits is not equal.

4. Define a macro EQUALSTR which compares two strings x and y and gives 1 if equal and 0 otherwise. Use this macro to accept two strings from the user and check if they are equal.

Signature of the instructor

Date  /  /

### Set B . Write C programs for the following problems.

1. Write a program to accept two strings as command line arguments and display the union and intersection of the strings. If the user enters invalid number of arguments, display appropriate message.

2. Write a program which accepts a string and an integer (0 or 1) as command line arguments. If the integer entered is 0, sort the string alphabetically in the ascending order and if it is 1, sort it in the descending order. If the user enters invalid number of arguments, display appropriate message. (Hint – use atoi)

Signature of the instructor

Date  /  /

### Set C . Write C programs for the following problems.

1. Create a header file “mymacros.h” which defines the following macros.
  - i. SQR(x) ii. CUBE(x) - nested iii. GREATER2(x,y) iv. GREATER3 (x,y,z) – nested
  - v. FLAG ( value = 1) (which may or may not be defined)

Include this file in your program. Write a menu driven program to use macros SQR, CUBE, GREATER2 and GREATER3. Your program should run the first two macros if the macro called FLAG has been defined. If it is not defined, execute the other two macros. Run the program twice – with FLAG defined and with FLAG not defined.

Signature of the instructor

Date  /  /

### Assignment Evaluation

### Signature

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

## Exercise 18

Start Date

/ /



To demonstrate text files using C



You should read the following topics before starting this exercise

1. Concept of streams
2. Declaring a file pointer
3. Opening and closing a file
4. Reading and Writing to a text file
5. Command line arguments



Operations performed	Syntax	Example
Declaring File pointer	FILE * pointer;	FILE *fp;
Opening a File	fopen("filename",mode); where mode = "r", "w", "a", "r+", "w+", "a+"	fp=fopen("a.txt", "r");
Checking for successful open	if (pointer==NULL)	if(fp==NULL) exit(0);
Checking for end of file	feof	if(feof(fp)) printf("File has ended");
Closing a File	fclose(pointer); fcloseall();	fclose(fp);
Character I/O	fgetc, fscanf fputc, fprintf	ch=fgetc(fp); fscanf(fp, "%c",&ch); fputc(fp,ch);
String I/O	fgets, fscanf fputs, fprintf	fgets(fp,str,80); fscanf(fp, "%s",str);
Reading and writing formatted data	fscanf fprintf	fscanf(fp, "%d%s",&num,str); fprintf(fp, "%d\t%s\n", num, str);
Random access to files	ftell, fseek, rewind	fseek(fp,0,SEEK_END); /* end of file*/ long int size = ftell(fp);

### Sample Code 1

The following program reads the contents of file named a.txt and displays its contents on the screen with the case of each character reversed.

/\* Program reverse case of characters in a file \*/

```
#include <stdio.h>
#include <ctype.h>
void main()
{
    FILE * fp;
    fp = fopen("a.txt", "r");
    if(fp==NULL)
    {
        printf("File opening error");
    }
}
```

```

        exit(0);
    }
    while( !feof(fp))
    {
        ch = fgetc(fp);
        if(isupper(ch))
            putchar(tolower(ch));
        else
            if(islower(ch))
                putchar(toupper(ch));
            else
                putchar(ch);
    }
    fclose(fp);
}

```

### Sample Code 2

The following program displays the size of a file. The filename is passed as command line argument.

```

/* Program to display size of a file */

#include <stdio.h>
void main(int argc, char *argv[])
{
    FILE * fp;
    long int size;
    fp = fopen(argv[1], "r");
    if(fp==NULL)
    {
        printf("File opening error");
        exit(0);
    }
    fseek(fp, 0, SEEK_END); /* move pointer to end of file */
    size = ftell(fp);
    printf("The file size = %ld bytes", size);
    fclose(fp);
}

```

### Sample Code 3

The following program writes data (name, roll number) to a file named student.txt , reads the written data and displays it on screen.

```

#include <stdio.h>
void main()
{
    FILE * fp;
    char str[20]; int num;
    fp = fopen("student.txt", "w+");
    if(fp==NULL)
    {
        printf("File opening error");
        exit(0);
    }
    fprintf(fp, "%s\t%d\n", "ABC", 1000);
    fprintf(fp, "%s\t%d\n", "DEF", 2000);
    fprintf(fp, "%s\t%d\n", "XYZ", 3000);
}

```



```

rewind(fp);
while( !feof(fp))
{
    fscanf(fp,"%s%d", str, &num);
    printf("%s\t%d\n", str, num);
}
fclose(fp);
}

```



1. Create a file named a.txt using the vi editor. Type the sample program 1 given above and execute the program. Modify the program to accept a character from the user and count the total number of times character occurs in the file.

2. Type the sample program 2 above and execute it. Modify the program to display the last n characters from the file.

3. Type the sample program 3 above and execute it. Modify the program to accept details of n students and write them to the file. Read the file and display the contents in an appropriate manner.

Signature of the instructor

Date  /  /



### Set A . Write C programs for the following problems.

1. Write a program to accept two filenames as command line arguments. Copy the contents of the first file to the second such that the case of all alphabets is reversed.

2. Write a program to accept a filename as command line argument and count the number of words, lines and characters in the file.

3. Write a program to accept details of n students (roll number, name, percentage) and write it to a file named "student.txt". Accept roll number from the user and search the student in the file. Also display the student details having the highest percentage.

Signature of the instructor

Date  /  /

### Set B. Write programs to solve the following problems

1. A file named numbers.txt has a set of integers. Write a C program to read this file and convert the integers into words and write the integer and the words in another file named numwords.txt.

Example:

numbers.txt

numwords.txt

11

Eleven

261

Two hundred Sixty One

9

Nine

2. Write a program which accepts a filename and an integer as command line arguments and encrypts the file using the key. (Use any encryption algorithm)

Signature of the instructor

Date  /  /

**Set C . Write C programs for the following problems.**

1. A text file contains lines of text. Write a program which removes all extra spaces from the file.
2. Write a menu driven program for a simple text editor to perform the following operations on a file, which contains lines of text.
  - i. Display the file
  - ii. Copy m lines from position n to p
  - iii. Delete m lines from position p
  - iv. Modify the nth line
  - v. Add n lines
3. Write a program which reads the contents of a C program and replaces all macros occurring in the program with its value. Assume only simple substitution macros (ex: #define FALSE 0 ).

Signature of the instructor

Date  /  /

**Assignment Evaluation**

**Signature**

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>

## Exercise 19

Start Date

/ /



**To demonstrate binary file handling using C.**



You should read the following topics before starting this exercise

1. Concept of streams
2. Declaring a file pointer
3. Opening and closing files
4. File opening modes
5. Random access to files
6. Command line arguments



In binary files, information is written in the form of binary . All data is written and read with no interpretation and separation i.e. there are no special characters to mark end of line and end of file.

I/O operations on binary files

Reading from a binary file	fread(address,size-of-element,number of elements,pointer);	fread (&num,sizeof(int),1,fp); fread (&emp,sizeof(emp),1,fp); fread(arr,sizeof(int),10,fp);
Writing to a binary file	fwrite(address,size-of-element,number of elements,pointer);	fwrite (&num,sizeof(int),1,fp); fwrite (&emp,sizeof(emp),1,fp);

### Sample Code

```
/* Program to demonstrate binary file */

struct employee
{
    char name[20];
    float sal;
};
main( )
{
    FILE *fp;
    struct employee e;
    int i;
    if((fp=fopen ("employee.in","wb"))==NULL)
    {
        printf("Error opening file");
        exit( );
    }

    for(i=0;i<5;i++)
    {
        printf("\n Enter the name and salary");
        scanf("%s%f",e.name,&e.sal);
        fwrite(&e,sizeof(e),1,fp);
    }
}
```

```

fclose(fp);

fp=fopen("employee.in","rb"); /*      reopen file */
if(fp==NULL)
{
    fprintf(stderr, "Error opening file);
    exit( );
}
for(i=0;i<5;i++)
{
    fread(&e,sizeof(e),1,fp);
    printf("\n Name = %s Salary = %f",e.name,e.sal);
}
fclose(fp);
}

```



1. Type program given above, writes data of 5 employees to a binary file and then reads the file. Modify the program to search an employee by name.

Signature of the instructor

Date  /  /



### Set A . Write C programs for the following problems.

1. Create a structure student (roll number, name, percentage) Write a menu driven program to perform the following operations on a binary file- "student.dat". Write separate functions for the different options.

1. Add a student (Note: Students should be assigned roll numbers consecutively)
2. Search Student
  - a. according to name
  - b. according to roll number
3. Display all students

2. Create a structure student (roll number, name, percentage) Write a menu driven program to perform the following operations on a binary file- "student.dat". Write separate functions for the different options.

1. Add a student (Note: Students will be assigned roll numbers consecutively)
2. Modify details
  - a. according to name
  - b. according to roll number
3. Display all students

3. Create a structure student (roll number, name, percentage). Write a menu driven program to perform the following operations on a binary file- "student.dat". Write separate functions for the different options.

1. Add a student (Note: Students will be assigned roll numbers consecutively)
2. Delete student
  - a. according to name
  - b. according to roll number
3. Display all students

Signature of the instructor

Date  /  /

**Set B . Write C programs for the following problems.**

1. Create two binary files such that they contain roll numbers, names and percentages. The percentages are in ascending orders. Merge these two into the third file such that the third file still remains sorted on percentage. Accept the three filenames as command line arguments.

2. Create a structure having the following fields:

Structure name: \_\_\_\_\_

Fields: \_\_\_\_\_

Store information for n variables of the above structure in a binary file. Write a menu driven program to perform the following operations Write separate functions for the different options.

i) \_\_\_\_\_ ii) \_\_\_\_\_ iii) \_\_\_\_\_ iv) \_\_\_\_\_

Signature of the instructor

Date  /  /

**Set C . Write C programs for the following problems.**

1. Create a binary file which contains details of student projects namely roll number, project name, project guide. The first line of the file contains an integer indicating the total number of students. When the program starts, read all these details into an array and perform the following menu driven operations. When the user selects Exit from the menu, store these details back into the file.

1. Add                      2. Delete                      3. Search                      2. Modify                      3. Display all                      4. Exit

Signature of the instructor

Date  /  /

**Assignment Evaluation**

**Signature**

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

## Exercise 20

Start Date

/ /



### Problem Solving Assignment



**Write C programs for the following problems.**

#### 1. The calendar problem

Display a calendar for a particular year. If month-number is supplied, only that month is displayed.

---

#### 2. Large number multiplication problem

Write a program that will multiply two (2) N digit positive integers, where N may be arbitrarily large. Your program should output the product(s).

**Input format:** One N digit positive integer per line in input and output files.

**Sample Input:**

```
65656432310964579864321356898765432243578987876654
94454
```

**Sample Output:**

```
6201512657499848426504609444515990137135009720901476916
```

---

#### 3. Room IDentification Problem

A company has just finished construction of their new pentagon shaped office building. However identifying the location of a room is a problem.

Here is how the room numbering scheme works:

Room numbers will be between 100 and 99999 inclusive. The ones digit (rightmost) tells which side of the pentagon the room is on. The next one after that, the tens digit, gives the hall number in which the room is (see table). The digits after that give which floor the room is on.

0 , 1	Hall 1
2 , 3	Hall 2
4 , 5	Hall 3
6 , 7	Hall 4
8 , 9	Hall 5

The least significant digit (right-most) tells whether the room is on the courtyard or outside edge of the hall. If it is even, the room faces the courtyard, if it is odd, it faces the outside.

**Input Format:** The input will consist a list of room numbers not longer than 5 digits. The input ends with -1.

**Output Format:** You will print the Room, Floor, Hall, and Side that each room number represents.

Room  $r$  is on Floor  $f$  in Hall  $h$  facing {*courtyard*|*outside*}

**Example Input:**

111  
1322  
455  
512  
-1

**Example Output:**

Room 111 is on Floor 1 in Hall 1 facing outside  
Room 1322 is on Floor 13 in Hall 2 facing courtyard  
Room 455 is on Floor 4 in Hall 3 facing outside  
Room 512 is on Floor 5 in Hall 1 facing courtyard

---

#### 4.The Anagram Problem

An anagram is a pair of words or sentences that contain the same number of the same letters. Examples include Dormitory whose anagram is Dirty Room. You will write a program to recognize whether a pair of words or sentences are anagrams.

**Input:** The input accept two strings and check whether they are anagrams.

**Output:** If a pair of strings tested are anagrams of each other, print "An Anagram," otherwise print, "Not An Anagram."

**Example Input 1:**

dormitory  
dirtyroom

**Output:**

An Anagram

**Example Input 2:**

eleven plus two  
twelve plus one

**Output:**

An Anagram

**Example Input 3:**

thisisntananagram  
andthatissuchashame

**Output:**

Not An Anagram

---

#### 5. The Secret Word problem :

You have determined that the enemy is using the following mechanism to encode secret words. You believe that the first letters of each word in enemy messages form secret words. Only the first letters of consecutive words are used to form the secret words. Further, a sentence may contain other words before and/or after the actual words that make up the secret word. Messages always contain a single space between words.

Write a program that takes a secret word and a message as input and determines if the message contains the secret word. The program should not be case-sensitive and should ignore punctuation.

**Input Format:** The first line of input consists of the secret word. The second line contains the sentence to check.

**Output Format:** The output will be "Secret word found" if the secret word is found in the sentence, otherwise, output "Secret word not found."

**Input:**  
year  
the yellow elephant ate raw bananas

**Output:**  
Secret word found

**Input:**  
you  
you will often fail unless you try harder

**Output:**  
Secret word not found

## 6. The Credit Card Verification Problem

You are provided with a credit card number with the length varying from 13 digits to 16 digits. Each digit of the credit card is weighted by either 2 or 1. The credit card number must be zero-filled on the left to create a sixteen digit number, and then the pattern starts with 2, alternating with a 1. If the number multiplied by the weight results in a 2-digit number, each digit is added to the sum. The final sum with the check digit should be a multiple of 10.

Example:

5	4	9	9	0	0	1	1	0	0	1	2	0	0	3	4	
2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	
1+0	+4	+1+8	+9	+0	+0	+2	+1	+0	+0	+2	+2	+0	+0	+6	+4	=40

$40 \bmod 10 = 0$ . If the last digit did not result in a number divisible by 10, the credit card number is invalid.

**Input Format:** The user will provide you with a credit card number without spaces.

**Output Format:** The program will return "Valid" or "Invalid" depending on the success or failure of the check digit.

**Input:**  
5499001100120034

**Output:**  
Valid

**Input:**  
5499001100120036

**Output:**  
Invalid

## 7. The library problem:

Write a program to solve the following problem: A library wants a program that will calculate the due date for a book on the basis of the issue date. The no. of days for which the book is issued is decided by the librarian at the time of issuing the book. For e.g. If the librarian enters the current date as 14-01-2000 and the no of days in which the book is due as 15, then your program should calculate the due date and give the output as 29-01-2000. Your program should accept the



current date (day, month, year) and the number of issue days as input and generate the due date as output.

---

### 8. The Comment Removal problem

Write a C program which reads the contents of a file containing a C program and removes all comments from the program.

---

### 9. The Histogram problem

Write a C program which reads the contents of a text file and generates a histogram of frequencies of all alphabets in the file. Use \* to draw the histogram bars.

---

### 10. The Cryptarithmic puzzle

“Cryptarithmic” puzzles are puzzles in which one gets problems like these

```
hello
+ there
-----
world
```

and is asked to assign digits to each letter so that the resulting addition is correct. Each digit from 0 to 9 must be used at most once, and the leading digits may not be 0. In the above cases, for example, we can get the solutions

```
56442
+ 15606
-----
72048
```

Write a program to find a solution to cryptarithmic problems for which the input consists of triples of strings each containing up to 128 lower-case letters and the output is in the form given in the sample below.

For the input  
hello there world

Produce the output

```
hello      56442
+ there    + 15606
-----
world      72048
```

---

### 11. The Compression problem

Write a program which compresses a text file such that consecutive occurrences of specific character are replaced by the character followed by a digit indicating the number of times the character occurs. Replace only if the character occurs 3 or more times consecutively. For example, if the input text is “aath1111yy66666kkk baabbbbdg”, the output should be “aath14yy65k3 4baab4dg”. Write a decompression program which reads a compressed file and generates the original text.

---

### 12. The 4 queens problem

This problem is to place 4 queens on a 4X4 chessboard such that no two queens can attack. i.e. No two queens are on the same row, same column or diagonal. Write a program to generate all possible valid placements. One possible solution is shown below.

	Q		
			Q
Q			
		Q	

The output is a set of column numbers  $\{c_1, c_2, c_3, c_4\}$  such that  $c_j$  is the column number in which Queen  $j$  is placed (in row  $j$ ). For the above example, the output is  $\{2, 4, 1, 3\}$ . Extend your program for  $n$  queens.

Signature of the instructor

Date  /  /

### Assignment Evaluation

### Signature

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>