21CSS101J – PROGRAMMING FOR PROBLEM SOLVING LABORATORY

SEMESTER II

ACADEMIC YEAR: 2022-2023

NAME :

REG.NO:

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under SECTION 3 of the UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603203.
CHENGALPATTU DISTRICT

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

BONAFIDE CERTIFICATE

Register No.:

Certified to be the bonafide record of work done by ___________________________________of ________________ **B.Tech.** Degree course in the Practical ____ in **SRM Institute of Science and Technology**, Kattankulathur during the academic year 2022-2023

Staff In-Charge

Date: **Academic Advisor**

LIST OF EXPERIMENTS

Ex. No.	Name of the Experiment
1	Input, Output Statements, Variables
2	Data types & Operators
3	Control Statements (Branching, Looping)
4	Two dimensional Arrays
5	Arrays with Pointers
6	Strings
7	Functions
8	Arrays and Functions
9	Input, Output in Python
10	Python and data structures
11	Arrays in Python

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 1					
Experiment Title: INPUT, OUTPUT STATEMENT, VARIABLES					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No.: 1 **INPUT, OUTPUT STATEMENT, VARIABLES**

Aim:

To write a program for input, output statements and variables using C

Theory

Input and Output Functions

The printf() Function

The use of printf function for printing captions and numerical results. The printf function provides certain features that can be effectively exploited to control the alignment and spacing of print-outs on the terminals. The general for of printf statement is

printf("control string",arg1,arg2,…,arg-n);

control string consists of three types of items:

1. Characters that will be printed on the screen as they appear.
2. Format specifications that define the output format for display of each item.
3. Escape sequence characters such as \n, \t, and \b.

The control string indicates how many arguments follow and what their types are. The arguments arg1,arg2,…,arg-n are the variables whose values are formatted and printed according to the specifications of the control string.

A simple format specification has the following form:

%w.p type-specifier

where w specifies the total number of columns for the output value and p specifies the numberof digits to the right of the decimal point or the number of characters to be printed from a string.

Example:

```
printf("Programming in C");printf("
");
printf("\n");
printf("%d",x);
printf("a=%f\nb=%f",a,b);
printf("sum = %d",1234);
printf("\n\n");
```

The scanf() Function

The scanf function can be used to read formatted input. The general form of scanf is

```
scanf("control string",arg1,arg2,...,arg-n);
```

Example:

scanf("%d%c%f%s",&count,&code,&ratio,name);

Commonly used formatting codes are given in the following table.

CODE	MEANING
%c	Read a single character
%d	Read a decimal integer
%e	Read a floating point value
%f	Read a floating point value
%g	Read a floating point value
%h	Read a short integer
%i	Read a decimal, hexadecimal, or octal integer
%o	Read an octal integer
%s	Read a string
%u	Read an unsigned decimal integer
%x	Read a hexadecimal integer

Problem Description 1:

Tina's brother gave her a friendly task of calculating the number of squares in a board that has n*n squares of dimensions 1cm *1cm each. Help her to find the number of total squares including all small and big ones.

Constraints: $2 \leq n \leq 20$

Input Format: The only line of the input represents a value of "n"

Output Format: Print the number of squares in the n*n board."

Problem Description 2:

Binita was travelling from Chennai to Delhi in Rajdhani Express. The train have arrived at the destination later than the estimated time. So, Binita wants to know the total number of hours and minutes the train was delayed. Can you help Binita in finding the exact hour and time Rajdhani Express was delay on the day of Binita's journey?

Constraint: $100 \leq$ tot_mins ≤ 550

Input Format: The only line of input has single value of variable tot_mins of type integer representing total minutes.

Output Format: Print the Number of Hours and Minutes in a single line.

Result :

Thus the programs are written using input, output statements, variables in C and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 2					
Experiment Title: DATA TYPES AND OPERATORS					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No. 2: DATA TYPES AND OPERATORS

Aim

To write a C program using different data types and operators

Theory

An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. C operators can be classified into a number of categories. They include

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators.

Arithmetic Operators

C provides all the basic arithmetic operators. They are listed in the following table:

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

Relational Operators

The relational operators are used to compare two values. An expression containing a relational operator is termed as a relational expression. The value of a relational expression is either one or zero. C supports six relational operators. These operators and their meanings are shown in the following table

Operator	Meaning
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Is equal to
!=	Is not equal to

== and != relational operators are also known as equality operators.

Logical Operators

In addition to the relational operators, C has three logical operators. They are

Operator	**Meaning**
&&	Logical AND
||	Logical OR
!	Logical NOT

Assignment Operators

Assignment operators are used to assign the result of an expression to a variable. The assignment operator is '='. Assignment expressions that make use of this operator are written in the form

identifier = expression;

Where identifier generally represents a variable and expression represents a constant, a variableor a more complex expression.

Example:

```
a = 3;
x = y;
```

Increment and Decrement Operators

The increment and decrement operators are

++ and --

The increment operators causes its operand to be increased by 1, whereas the decrement operator causes its operand to be decreased by 1.

Conditional Operator (Ternary Operator)

A ternary operator pair "?:" is available in C to construct conditional expressions of the form

exp1 ? exp2 : exp3;

Where exp1, exp2, and exp3 are expressions.

Here exp1 is evaluated first. If it is nonzero (true), then the expression exp2 is evaluatedand becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.

Example:

```
c=(a>b)?a:b;
```

Bitwise Operators

The bitwise operators are used for testing the bits, or shifting them right or left. Bitwise operators may not be applied to float or double. The following table lists the bitwise operators and their meanings.

Operator	**Meaning**
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right
~	One’s complement

Special Operators

C supports some special operators. They are

- comma operator
- sizeof operator
- pointer operators (& and *)
- member selection operator (. and ->)

The comma operator

The comma operator can be used to link the related expressions together.

Examples:

1. The statement

```
value = (x = 10, y=5, x+y);
```

first assigns the value 10 to x, then assigns 5 to y, and finally assigns 15 to value.

2. In for loops: `for(n=1,m=10;n<=m;n++,m++)`

3. In while loops: `while(c=getchar(),c!=’\0’)`

4. Exchanging values: `t=x,x=y,y=t;`

The sizeof operator

The sizeof is a compile time operator and returns the number of bytes the operand occupies.

Example:

```
m=sizeof(sum);
n=sizeof(long int);
k=sizeof(234L);
```

Problem Description1:

Arav and Aaron are participating in the Bike racing. Arav crossed some milestones earlier and Aaron crossed some milestones earlier during their racing because they have changed their speeds at different times.Both of them like to know the difference in speeds between them at different stages of racing.Can you help find the speed difference between Arav and Aaron?

Constraints: $20 \leq$ aravspeed ≤ 100 , $20 \leq$ aaronspeed ≤ 100

Input Format: The first line of input represents the speed of Arav. The second line of input represents speed of Aaron.

Output Format: Print difference between the driving speed of two participants in a single line.

Problem Description2:

When the wind blows in cold weather, the air feels even colder than it actually is because the movement of the air increases the rate of cooling for warm objects, like people. This effect is known as wind chill.

In 2016, Jammu and Kashmir, Delhi, and Himachal Pradesh adopted the following formula for computing the wind chill index. Within the formula, **Ta** is the air temperature in degrees celsius and **V** is the wind speed in kilometers per hour.

Constraint:

- Ta and v are float values.

Formulation to find wind chill index is:

$$WCI = 13.12 + 0.6215*Ta - 11.37*V^{0.16} + 0.3965*Ta* V^{0.16}$$

Result :

Thus the programs are written using different data types and operators in C and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 3					
Experiment Title: Control Statements (Branching, Looping)					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No. 3: **CONTROL STATEMENTS (BRANCHING, LOOPING)**

Aim

To write a C program using different control statements

Theory

Decision Making and Branching

C language supports control or decision making statements. They are

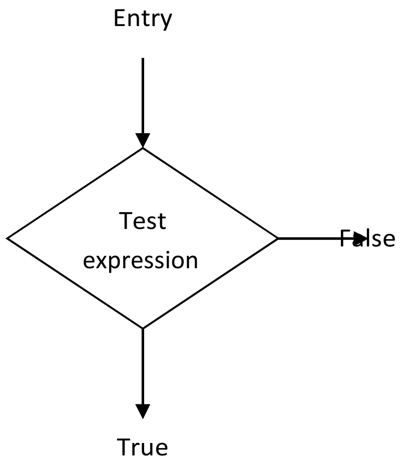
1. if statement
2. switch statement
3. Conditional operator statement
4. goto statement

Decision making with if statement

The if statement is a powerful decision making statement and is used to control the flowof execution of statements. The syntax of if statement is

if (condition)

It allows the computer to evaluate the condition first and then, depending on whether the value of the condition is 'true' (non-zero) or 'false' (zero), it transfers the control to a particular statement. This point of program has two paths to follow, one for the true condition and the other for the false condition as shown in the following fig.



Types of if statements

1. Simple if statement
2. if….else statement

3. Nested if….else statement
4. else if ladder

Simple if Statement

The general form of a simple if statement is

```
if (condition)
 {
   Statement-block;
 }
Statement-x;
```

The statement-block may be single statement or a group of statements. If the condition is true, the statement-block will be executed; otherwise the statement-block will be skippedand the execution will jump to the statement-x.

The if…else Statement

The if…else statement is an extension of the simple if statement. The general form is

```
if (condition)
  {
    True-block Statement(s);
  }
else
  {
    False-block Statement(s);
  }
 Statement-x;
```

If the condition is true, then the true-block statement(s), immediately following the ifstatement are executed; otherwise, the false-block statement(s) are executed.

Nesting of if…else Statement

When a series of decisions are involved, we may have to use more than one if…elsestatement in nested form as follows:

```
if (condition 1)
  {
    if (condition 2)
      {
        Statement-1;
      }
    else
      {
        Statement-2;
      }
  }
else
  {
    Statement-3;
  }
 Statement-x;
```

If the condition-1 is false, the statement-3 will be executed; otherwise it continues to perform the second test. If the condition-2 is true, the statement-1 will be evaluated; otherwise the statement-2 will be evaluated and then the control is transferred to the statement-x.

The else…if ladder (if…else if statement)

```
if (condition 1)
  Statement-1;
else if (condition 2)
       Statement-2;
     else if (condition 3)
            Statement-3;
             ………….
          else if (condition n)
               Statement-n;
               else
                 Default-Statement;
Statement-x;
```

A multipath decision is a chain of ifs in which the statement associated with each elseis an if. The general form of if…else if statement is

This construct is known as the else if ladder. The conditions are evaluated from the top, downwards. As soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement-x. When all the n conditions become false, then the final else containing the default-statement will be executed.

The switch Statement

The switch statement is a built-in multiway decision statement. The switch statement tests the value of a given variable against a list of case values and when a match is found, a block of statements associated with that case is executed. The general form of the switch statement is as shown below:

```
                switch(expression)
   {
     case value-1:
                          block-1;
                          break;
     case value-2:
                          block-2;
                          break;
       ……
       ……
       ……
     case value-n:
                          block-n;
                          break;
     default:
                          default-block;
                          break;
   }
```

The expression is an integer expression or characters. Value-1, value-2,… are constants or constant expressions and are known as case labels. Each of these values should be unique within a switch statement. Block-1, block-2,… are statement lists and may contain zero or more statements.

When the switch is executed, the value of the expression is successively compared against the values value-1, value-2,… if a case is found whose value matches with the value of the expression, then the block of statements that follows the case are executed.

The break statement at the end of each block signals the end of a particular case and causes an exit from the switch statement.

The default is an optional case. When present, it will be executed if the value of the expression does not match with any of the case values.

The goto statement

The goto statement is used to branch unconditionally from one point to another in the program. The goto requires a label in order to identify the place where the branch is to be made. A label is any valid variable name, and must be followed by a colon. The label is placedimmediately before the statement where the control is to be transferred. The general forms of goto and label statements are shown below:

```
goto label;
    …
    …
    …
label:
Statement;
```

Forward jump

```
label:
Statement;
    …
    …
    …
goto label;
```

Backward jump

The label: can be anywhere in the program either before or after the goto label; statement. If the label: is before the statement goto label; a loop will be formed and some statements will be executed repeatedly. Such a jump is known as a backward jump. On the other hand, if the label: is placed after the goto label; some statements will be skipped and the jump is known as a forward jump.

Decision Making and Looping

Loop is a sequence of statements executed until some conditions for termination of the loop are satisfied. A program loop consists of two segments, one known as the body of theloop and the other known as the control statement. The control statement tests certain conditions and then directs the repeated execution of the statements contained in the body ofthe loop.

Depending on the position of the control statement in the loop, a control structure may be classified either as the entry-controlled loop or as the exit-controlled loop.

The C language provides for three loop constructs for performing loop operations. Theyare:

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

The while Statement

The simplest of all the looping structures in C is the while statement. The while is an entry-controlled loop statement. The condition is evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the condition is once again evaluated and if it is true, the body is executed once again. This process of repeated execution of the body continues until the condition finally becomes false and the control is transferred out of the loop. On exit, the program continues with the statement immediately after the body of the loop. The basic format of the while statement is

```
while (condition)
 {
   body of the loop
 }
```

The body of the loop may have one or more statements. The braces are needed only ifthe body contains two or more statements.

The do…while Statement

In do…while statement, the body of the loop is evaluated first. At the end of the loop, the condition in the while statement is evaluated. If the condition is true, the program continuesto evaluate the body of the loop once again. This process continues as long as the condition is true. When the condition becomes false, the loop will be terminated and the control goes to the statement that appears immediately after the while statement. The general format of the do…while statement is

```
do
  {

    body of the loop

  } while (condition);
```

Since the condition is evaluated at the bottom of the loop, the do…while construct provides anexit-controlled loop and therefore the body of the loop is always executed at least once.

The for Statement

The for loop is another entry-controlled loop that provides a more concise loop controlstructure. The general form of the for loop is

```
for(initialization; condition; increment)
  {
        body of the loop
  }
```

The execution of the for statement is as follows:

1. Initialization of the control variables is done first. Eg. i=1 and count=0.
2. The value of the control variable is tested using the condition. If the condition is true, the body of the loop is executed; otherwise the loop is terminated and the execution continues with the statement that immediately follows the loop.
3. When the body of the loop is executed, the control is transferred back to the for statement after evaluating the last statement in the loop. Now, the increment section is executed and the condition is evaluated. If the condition is satisfied, the body of the loop is again executed. This process continues till the value of the control variable fails to satisfy the condition.

Problem Description:

Laaysa with her friends going to the theatre for a movie. The seating arrangement is triangular in shape. Theatre staff insisted the audience to sit in odd row if the seat number is odd and in even row if the seat number is even. But the instruction is very confusing for Laaysa and her friends.So help them with the seating layout so that they can sit in correct seats.

Note: Do use the printf() function with a newline character (\n).

Constraints: $4 \leq N \leq 20$

Input Format: Only line of input has single integer value representing the number of rows in the theatre.

Output Format: Print the layout based on the number of rows specified in input.

Instruction: To run your custom test cases strictly map your input and output layout with the visible test cases

Result:

Thus the programs are written using different control statements in C and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 4					
Experiment Title: ARRAYS					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No. 4: ARRAYS

Aim

To write a C program for a group of related data items that share a common name using array.

Theory

One-Dimensional Arrays

A list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable or a one-dimensional array. The subscript can begin with number 0. That is a[0] is allowed. For example, if we want to represent a set of five numbers by an array variable number, then we may declare the variable no as follows

int no[5];

and the computer reserves five storage locations as shown below:

	no[0]
	no[1]
	no[2]
	no[3]
	no[4]

The values to the array elements can be assigned as follows:

no[0]=37
no[1]=98
no[2]=57
no[3]=35
no[4]=71

This would cause the array no to store the values as shown below:

37	**no[0]**
98	**no[1]**
57	**no[2]**
35	**no[3]**
71	**no[4]**

Declaration of Arrays

Arrays must be declared before they are used. The general form of array declaration is

type variable-name[size];

The type specifies the type of element that will be contained in the array, such as int, float, or char and the size indicates the maximum number of elements that can be stored inside the array. For example,

float height[25];

declares the height to be an array containing 25 real elements. Any subscripts 0 to 24 are valid.

Two-Dimensional Arrays

C allows us to define a table of items by using two-dimensional arrays. The two-dimensional arrays are declared as follows:

type array_name[row_size][column_size];

Two-dimensional arrays are stored in memory as shown in the following fig.

	Column 0	**Column 1**	**Column 2**
Row 0	**0,0**	**0,1**	**0,2**
Row 1	**1,0**	**1,1**	**1,2**
Row 2	**2,0**	**2,1**	**2,2**
Row 3	**3,0**	**3,1**	**3,2**

Each dimension of the array is indexed from zero to its maximum size minus one; the first index selects the row and the second index selects the column within that row.

Problem Description:

Write a program in C to sort the matrix row-wise and column-wise.

Input format: The first line contains two integers m and n, representing the no.of rows and columns in the matrix respectively. The next m lines contain n space-separated integers in each line.

Output format; Print the sorted matrix.

Result :

Thus the programs is written for a group of related data items that share a common name in C and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 5					
Experiment Title: **ARRAYS WITH POINTERS**					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No. 5: **ARRAYS**

Aim

To write a C program using arrays with pointers

Theory

Pointers

A pointer is a variable whose value is the address of another variable, i.e., direct addressof the memory location. Like any variable or constant, you must declare a pointer before usingit to store any variable address. The general form of a pointer variable declaration is:

```
type *var-name;
```

Here, type is the pointer's base type; it must be a valid C data type and var-name is the nameof the pointer variable. The asterisk * used to declare a pointer is the same asterisk used for multiplication. However, in this statement the asterisk is being used to designate a variable asa pointer.

Example:

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise,is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

There are a few important operations, which we can do with the help of pointers veryfrequently.
(a) Define a pointer variable
(b) Assign the address of a variable to a pointer and
(c) Access the value at the address available in the pointer variable.

This is done by using unary operator * that returns the value of the variable located at theaddress specified by its operand. The following example makes use of these operations −

Problem Description:

Program to find the number of prime numbers and number of composite numbers in an array to which memory is allocated dynamically using pointers.

Input format: The first line contains an integer n, representing the number of elements in that array. The next line contains an integer array **arr[]** with n space-separated values.

Output format: The first line contains a number of primes The second line contains a number of composites

Result :

Thus the program to find the number of prime numbers and number of composite numbers using pointers in C is written and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 6					
Experiment Title: STRINGS					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No. 6: ARRAYS

Aim

To write a C program using arrays with pointers

Theory

A string is an array of characters. Any group of characters defined between double quotation marks is a constant string.

Example:

“Welcome to C”

Character strings are often used to build meaningful and readable programs. The common operations performed on character strings are:

- Reading and Writing strings
- Combining strings together.
- Copying one string to another
- Comparing strings for equality
- Extracting a portion of a string.

Declaring and Initializing String Variables

The general form of declaration of a string variable is

char string_name[size];

The size determines the number of characters in the string-name.

Example:

```
char name[30];
char city[20];
```

when the compiler assigns a character string to a character array, it automatically supplies a null character (‘\0’) at the end of the string. Character arrays may be initialized when they are declared. C permits a character array to be initialized in either of the following two forms:

```
static char city[9] = “NEW YORK”;
static char city[9] = {‘N’,’E’,’W’,’ ‘,‘Y’,’O’,’R’,’K’,’\0’};
```

C also permits us to initialize a character array without specifying the number of elements. In such cases, the size of the array will be determined automatically, based on the number of elements initialized.

Example:

```
static char city[] = {‘N’,’E’,’W’,’ ‘,‘Y’,’O’,’R’,’K’,’\0’};
```

defines the array city as a nine element array.

Reading Strings from Terminal

Reading Words

The familiar input function scanf can be used with %s format specification to read in a stringof characters.

Example:

```
 char city[20];
 scanf("%s",city);
```

The problem with the scanf function is that it terminates its input on the first white space it finds. Therefore, if the following line of text is types in at the terminal,

```
NEW YORK
```

Then only the string "NEW" will be read into the array city, since the blank space after the word "NEW" will terminate the string.

Reading a Line of Text

An entire line of text can be read and stored in an array using gerchar() function. The readingis terminated when the newline character('\n') is entered and the null character is then insertedat the end of the string.

Writing Strings to Screen

The printf function with %s format is used to print strings to the screen. The format %s can be used to display an array of characters that is terminated by the null character.

Example:

```
printf("%s",name);
```

We can also specify the precision with which the array is displayed. For instance, the specification

%10.4

indicates that the first four characters are to be printed in a field width of 10 columns.

Arithmetic Operations on Characters

C allows us to manipulate characters the same way we do with numbers. Whenever a character constant or character variable is used in an expression, it is automatically converted into an integer value by the system. The integer value depends on the local character set of the system.

For example, if the machine uses the
ASCII representation, then,

```
x='a';
printf("%d",x)
```

will display the number 97 on the screen.

It is also possible to perform arithmetic operations on the character constants and variables. For example,

```
x=’z’-1;
```

is a valid statement. In ASCII, the value of ‘z’ is 122 and therefore, the statement will assignthe value 121 to the variable x. We may also use character constants in relational expressions. For example, the expression

```
ch>=’A’ && ch<=’Z’
```

would test whether the character contained in the variable ch is an upper-case letter. We canconvert a character digit to its equivalent integer value using the following relationship:

```
x=ch-‘0’;
```

where x is defined as an integer variable and ch contains the character digit, For example, letus assume that the ch contains the digit ’7’, Then,

```
x = ASCII value of ‘7’ – ASCII value of ‘0’
  = 55 – 48
  = 7
```

The C library supports a function that converts a string of digits into their integer values. Thefunction takes the form

```
x=atoi(string);
```

x is an integer variable and string is a character array containing a string of digits. Consider the following segment of a program:

```
int year;
char no[]=”2007”;
year=atoi(no);
```

no is a string variable which is assigned the string constant “2007”. The function atoi convertsthe string “2007” to its numeric equivalent 2007 and assigns it to the integer variable year.

String Handling Functions

C library supports a large number of string-handling functions that can be used to carry out many of the string manipulations. The most commonly used string handling functions are:

strcat()	- Concatenates two strings
strcmp()	- Compares two strings
strcpy()	- Copies one string over another
strlen()	- Finds the length of a string

strcat() Function

The strcat() function joins two strings together. It takes the following form:

```
strcat(string1,string2);
```

string1 and string2 are character arrays. When the function strcat is executed, string2 is appended to string1. The string at string2 remains unchanged. For example, consider the following three strings:

```
s1=”VERY”;
s2=”GOOD”;
strcat(s1,s2);
```

strcat function may also append a string constant to a string variable.

Example:

```
strcat(s1,”GOOD”);
```

C permits nesting of strcat functions. For example, the statement

```
strcat(strcat(s1,s2),s3);
```

concatenates all the three strings together. The resultant string is stored in s1.

strcmp() Function

The strcmp() function compares two strings identified by the arguments and has a value 0 if they are equal. If they are not, it has the numeric difference between the first nonmatching characters in the strings. It takes the form:

```
strcmp(string1,string2);
```

string1 and string2 may be string variables or string constants.

strcpy() Function

The strcpy() function works almost like a string-assignment operator. It takes the form

```
strcpy(string1,string2);
```

and assigns the contents of string2 to string1. string2 may be a character array variable or astring constant. For example, the statement

strlen() Function

The strlen() function counts and returns the number of characters in a string.

```
n=strlen(string);
```

where n is an integer variable which receives the value of the length of the string.

Problem Description:

Write a program to check whether the given two strings are equal or not without using string library functions.

At the time of execution, the program should print the message on the console as:
string1 :
For example, if the user gives the input as:
string1 : Kaveri
Next, the program should print the message on the console as:
string2 :
For example, if the user gives the input as:

string2 : Kaveri
then the program should print the result as:
equal
In the same way if the input is given as "Kaveri" and "kaveri", then the result will be "not equal".
Note: Do use the printf() function with a newline character (\n) at the end.

Result:

Thus the program to check whether the given two strings are equal or not without using string library functions in C is written and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 7					
Experiment Title: FUNCTIONS					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No. 7: **FUNCTIONS**

Aim

To write a C program using Functions

Theory

User-Defined Functions

C functions can be classified into two categories, namely, library functions and user- defined functions. Advantages of user-defined functions are

1. It facilitates top-down modular programming as shown in the following figure.
2. The length of a source program can be reduced by using functions at appropriate places.
3. It is easy to locate and isolate a faulty function for further investigations.
4. A function may be used by many other programs.

The Form of C Functions

The general form of function is

```
function-name(argument list)
argument declaration
{
   local variable declarations;
   executable statement1;
   executable statement2;
         ……….
         ………
   return(expression);
}
```

Argument List:

The argument list contains valid variable names separated by commas. The list mustbe surrounded by parentheses. The argument variables receive values from the calling function. Some examples of functions with arguments are:

quadratic(a,b,c)
square(x)

Return Values and their Types

A function may or may not send back any value to the calling function. If it does, it is done through the return statement. The return statement can take one of the following forms

return;
Or
return(expression);

Calling a Function

A function can be called by simply using the function name in a statement. Example:

```
main()
{
  int p; p=mul(5,2);
  printf("%d",p);
}
```

Recursion

Calling a function with in the same function is called recursion.

Example:

```
#include<stdio.h> #include<conio.h> main()
{

printf("\nWelcome to C");
main();
}
```

When executed, this program will produce an output something like this:

```
Welcome to C
Welcome to C
Welcome to C
Welcome t
```

Execution is terminated abruptly; otherwise the execution will continue indefinitely.

Problem Description:

Write a C program to demonstrate functions without arguments and with return value.

The written code is used to check whether the given number is a prime number or not.

Write the function prime().

Sample Input and Output:

```
Enter a number : 5

The given number is a prime number
```

Result:

Thus the program to demonstrate functions without arguments and with return value in C is written and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 8					
Experiment Title: ARRAYS AND FUNCTIONS					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
				Total	

Ex. No. 8: **ARRAYS AND FUNCTIONS**

Aim

To write a C program using Functions

Theory

In C programming, we can pass an entire array to functions, to pass the array declare a formal parameter in one of following three ways and all three declaration methods produce similar results because each tells the compiler that an integer pointer is going to be received. Similarly, you can pass multi-dimensional arrays as formal parameters.

Way-1

Formal parameters as a pointer:

```
void myFunction(int *param) {
  .
  .
  .
}
```

Way-2

Formal parameters as a sized array:

```
void myFunction(int param[10]) {
  .
  .
  .
}
```

Way-3

Formal parameters as an unsized array:

```
void myFunction(int param[]) {
  .
  .
  .
}
```

Problem Description:

Write a program to find the number of distinct elements in an unsorted array. (Do it without sorting the array)

Input Format: Input consists of two lines. The first integer in the first line corresponds to n, the number of elements in the array. The next n integers correspond to the elements in the array. Assume that the maximum value of n is 15.

Output Format: Output consists of a single integer that corresponds to the number of distinct elements in the array.

Result:

Thus the program to find the number of distinct elements in an unsorted array in C is written and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 9					
Experiment Title: INPUT, OUTPUT IN PYTHON					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
Total					

Ex. No. 9: **INPUT, OUTPUT IN PYTHON**

Aim

To write a python program using input and output Functions

Theory

Introduction to Python

The Python programming language was developed in the 1980's by Guido Van Rossum at Centrum Wiskunde & Informatica (CWI) in Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system.

Python is an Interpreted, Interactive, Object Oriented Programming Language. Python provides Exception Handling, Dynamic Typing, Dynamic Data Types, Classes, many built-in functions and modules. Python is used to develop both Web Applications and Desktop Applications. Python is used in Artificial Intelligence, Scientific Computing and Data Analytics. Python standard libraries support XML, HTML, Email, IMAP etc.. Python is used to control Firmware updates in Network Programming.

The instructions (called as source code) written in Python programming language can beexecuted in two modes:

1. **Interactive mode**: In this mode, lines of code is directly written and executed in the Python interpreter shell, which instantaneously provide the results.
2. **Normal or script mode**: In this mode the source code is saved to a file with .py extension, and the file is executed by the Python interpreter.

Input and Output Statements

In Python, the input() function is used to read input from the user. The syntax for input() function is :

```
input([prompt])
```

Here, the optional prompt string will be printed to the console for the user to input a value. Theprompt string is used to indicate the type of value to be entered by the user.

Reading string inputs

```
name = input("Enter your Name: ")
print("User Name:", name)
```

Output:

```
Enter your Name: Raj KumarUser
Name: Raj Kumar
```

To take only specified data type input from the user, mention the data type before the input.

Example:

```
a= int(input("Enter an integer :")
```

Output Statement:

The print() function is used to print the values to the standard output. It takes zero or more number of expressions separated by comma (,). The print() function converts those expressions into strings and writes the result to standard output which then displays the result on screen.

The general syntax of print() function is:

print(value1, value2..., sep = ' ', end = '\n', file = sys.stdout, flush = False)

where,

value1,value2...,value-n These are the actual data values to printed.

sep - This is the separator used between each value. If there is no separator, then by default 'whitespace' is taken.

end - This is the character which gets printed after all values have been printed. The newline character '\n' is the default.

file - This argument specifies where the output needs to be printed. The screen is the standard output by default

Example-1:

```
print("Hi", "Hello", "Python") # will print output as followsHi

Hello Python
```

Example-2:

```
a = 10
b = 50
print("A value is", a, "and", "B value is", b)          # will print output as followsA

value is 10 and B value is 50
```

The above print() statement consists of both strings and integer values.

More on print(...) function:

With comma (,) as a separator: When two or more strings are given as parameters to a print(...) function with a comma (,) as a separator, the strings are appended with a space and printed.

For example:

print("I", "Love", "Python")will

generate the output as

I Love Python

With space as a separator: When two or more strings are given as parameters to a print(...) function with space as a separator, the strings are appended without a space between them.

For example:

print("I" "Love" "Python")will

generate the output as

ILovePython

With repeat character (*): We can print a string n times by using the repeat character (*) as shown below:

print("ABC" * 3);

will generate the output as

ABCABCABC

Output Formatting in Python

1. Python uses C-style string formatting to create new, formatted strings. The % operator also called as 'string modulo operator' is used to format a set of variables enclosed in a "tuple" (a fixed size list), together with a format string, which contains normal text together with "argument specifiers", special symbols like %s and %d.

The general syntax for a format placeholder is:

%[flags][width][.precision]type

The following are some basic argument specifiers:

%s - String (or any object with a string representation, like numbers)
%d - Integers
%f - Floating point numbers
%.<number of digits>f - Floating point numbers with a fixed amount of digits for thedecimal part.
%x or %X - Integers in hexadecimal representation (uppercase/lowercase)
%o - Octal representation

Example 1: If we consider a single variable called 'name' with a username in it, and if we wouldlike to print a greeting to that user, then:

```
name = "Raj Kumar"
print("Good morning, %s!" % name) # This statement prints "Good morning, Raj Kumar!"
```

Output:

```
Good morning, Raj Kumar!
```

Example 2: If we consider more than one variable, we must use a tuple of those variables asshown below:

```
a = 10
b = 20
str = "Hello"
print("The value of a = %d, b = %d and str = %s" %(a, b, str))
```

Output:

```
The value of a = 10, b = 20 and str = Hello
```

Example 3:

```
print("Total number of votes: %4d, Women votes: %2d" %(480, 70))
```

Output:

Total number of votes: _480, Women votes: 70 # Observe 1 leading blank space left before thefirst value and no leading blank space before the second value

The first placeholder %4d is used for the first component of our tuple, i.e. 480. This number will be printed with 4 characters. As 480 consists only of three digits, the output is padded with 1 leading blank.

The second placeholder %2d is used for the second component of our tuple, i.e. 70. This number will be printed with 2 characters. As 70 consists two digits, the output does not have any leading blanks as both the blanks are occupied by the digits.

The output can also be presented in a more elegant way using the str.format()
This style of string formatting eliminates the usage of % operator (string modulo operator).

The general syntax of str.format() function is:

```
        {}{}.format(value1,  value2,...,valuen)
```

where,

{}{} are place holders created by the user in the string.

value1,value2,...,valuen They could be variables, integers, floating-point numbers, strings,characters etc.

Note:The number of values passed as arguments in .format(arg1,arg2..) method should alwaysbe equal to the number of placeholders {} created by the user in the string.

The following are some basic argument specifiers:

{} - String (or any object with a string representation, like numbers)

{0:<number of digits>d} - Integers
{0:.<number of digits>f} - Floating point numbers with a fixed amount of digits for thedecimal part.
{0:X} or {0:x} - Integers in hex representation (uppercase/lowercase)
{0:o} - Octal representation

Problem Description1:

Your task is to:

Take two integers a and b as input from the user.

Perform all arithmetic operations (+, -, *, /, **, %, //) on the given integer values (In given sequence).

And print to the console, the result of every arithmetic operation. (For printing the results please refer the visible sample test case)

Problem Description2:

Take two string inputs from the user x and y.

Your task is to:

Add two strings x and y and assign the resultant value to other variable z.

Print the values in x, y and z in the exact format.

Result:

Thus a python program using input and output Functions are written and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 10					
Experiment Title: PYTHON AND DATA STRUCTURES					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
Total					

Ex. No. 10: **PYTHON AND DATA STRUCTURES**

Aim

To write a python program using input and output Functions

Theory

Types of Data Structures in Python

Python has implicit support for Data Structures which enable you to store and access data. These structures are called List, Dictionary, Tuple and Set.

Python allows its users to create their own Data Structures enabling them to have full control over their functionality. The most prominent Data Structures are Stack, Queue, Tree, Linked List and so on which are also available to you in other programming languages.

Built-in Data Structures

As the name suggests, these Data Structures are built-in with Python which makes programming easier and helps programmers use them to obtain solutions faster.

Lists

Lists are used to store data of different data types in a sequential manner. There are addresses assigned to every element of the list, which is called as Index. The index value starts from 0and goes on until the last element called the positive index. There is also negative indexing which starts from -1 enabling you to access elements from the last to first.

Creating a list

To create a list, use the square brackets and add elements into it accordingly. If you do not pass any elements inside the square brackets, you get an empty list as the output.

```
my_list = [] #create
empty list
print(my_list)
my_list = [1, 2, 3, 'example', 3.132] #creating list with data
print(my_list)
Output:
[]
[1, 2, 3, ‘example’, 3.132]
```

Adding Elements

Adding the elements in the list can be achieved using the append(), extend() and insert() functions.

The append() function adds all the elements passed to it as a single element.The
extend() function adds the elements one-by-one into the list.
The insert() function adds the element passed to the index value and increase the size of the listtoo.

```
my_list = [1, 2, 3]
```

```
print(my_list)
my_list.append([555, 12]) #add as a single element
print(my_list)
my_list.extend([234, 'more_example']) #add as different elements
print(my_list)
my_list.insert(1, 'insert_example') #add element i
print(my_list)
```

```
Output:
[1, 2, 3]
[1, 2, 3, [555, 12]]
[1, 2, 3, [555, 12], 234, ‘more_example’]
[1, ‘insert_example’, 2, 3, [555, 12], 234, ‘more_example’]
```

Deleting Elements

To delete elements, use the del keyword which is built-in into Python but this does not return anything back to us. If you want the element back, you use the pop() function which takes the index value.To remove an element by its value, you use the remove() function.

Accessing Elements

Accessing elements is the same as accessing Strings in Python. You pass the index values and hence can obtain the values as needed.

```
my_list = [1, 2, 3, 'example', 3.132, 10, 30]
for element in my_list: #access elements one by one
   print(element)
print(my_list) #access all elements
print(my_list[3]) #access index 3 element
print(my_list[0:2]) #access elements from 0 to 1 and exclude 2
print(my_list[::-1]) #access elements in reverse
```

Other Functions:

The len() function returns to us the length of the list.
The index() function finds the index value of value passed where it has been encountered the first time.
The count() function finds the count of the value passed to it.
The sorted() and sort() functions do the same thing, that is to sort the values of the list. Thesorted() has a return type whereas the sort() modifies the original list.

Dictionary

Dictionaries are used to store key-value pairs. To understand better, think of a phone directory where hundreds and thousands of names and their corresponding numbers have been added. Now the constant values here are Name and the Phone Numbers which are called as the keys. And the various names and phone numbers are the values that have been fed to the keys.If you access the values of the keys, you will obtain all the names and phone numbers. So thatis what a key-value pair is. And in Python, this structure is stored using Dictionaries.

Creating a Dictionary
Dictionaries can be created using the flower braces or using the dict() function. You need to add the key-value pairs whenever you work with dictionaries.

Changing and Adding key, value pairs
To change the values of the dictionary, you need to do that using the keys. So, you firstly accessthe key and then change the value accordingly. To add values, you simply just add another key- value pair as shown below.

```
my_dict = {'First': 'Python', 'Second': 'Java'}
print(my_dict)
my_dict['Second'] = 'C++' #changing element
print(my_dict)
my_dict['Third'] = 'Ruby' #adding key-value pair
print(my_dict)
```

Output:

```
{‘First’: ‘Python’, ‘Second’: ‘Java’}
{‘First’: ‘Python’, ‘Second’: ‘C++’}
{‘First’: ‘Python’, ‘Second’: ‘C++’, ‘Third’: ‘Ruby’}
```

Deleting key, value pairs
To delete the values, use the pop() function which returns the value that has been deleted.
To retrieve the key-value pair, use the popitem() function which returns a tuple of the key andvalue.
To clear the entire dictionary, use the clear() function.

Accessing Elements
You can access elements using the keys only. You can use either the get() function or just pass the key values and you will be retrieving the values.

Tuple
Tuples are the same as lists are with the exception that the data once entered into the tuple cannot be changed no matter what. The only exception is when the data inside the tupleis mutable, only then the tuple data can be changed.

Creating a Tuple
Tuple can be created using parenthesis or using the tuple() function.

```
my_tuple = (1, 2, 3) #create tuple
print(my_tuple)
```

Output:

```
(1, 2, 3)
```

Sets

Sets are a collection of unordered elements that are unique. Meaning that even if the data is repeated more than one time, it would be entered into the set only once.

Creating a set

Sets are created using the flower braces but instead of adding key-value pairs, just pass valuesto it.

```
my_set = {1, 2, 3, 4, 5, 5, 5} #create set
print(my_set)
```

Output:

```
{1, 2, 3, 4, 5}
```

Adding elements

To add elements, use the add() function and pass the value to it.

```
my_set = {1, 2, 3}
my_set.add(4) #add element to set
print(my_set)
```

Output:

```
{1, 2, 3, 4}
```

Operations in sets

The different operations on set such as union, intersection and so on are shown below.

```
my_set = {1, 2, 3, 4}
my_set_2 = {3, 4, 5, 6}
```

Problem Description1:

Take a string input from the user. Reverse the first three characters and print the reversed string as an output. If the string does not consist at least three characters, it should print "Insufficient".

Problem Description2:

Create a tuple with integer elements. Write a program to find maximum element in the tuple.

Result:

Thus a python program using data structures are written and executed successfully.

Rubrics:

Register No.:		**Name:**		**Date:**	
Experiment No.: 11					
Experiment Title: ARRAYS IN PYTHON					
Component	**Max. Marks**	**Grading Rubrics**			**Marks Scored**
Program Completion	4	Programs completed successfully [4 Marks]	Partially completed [1-3 Marks]	Not completed [0 Marks]	
Program Explanation	3	Correctly explained [3 Marks]	Partially explained [1.5-2 Marks]	Not able to explain the code [0 marks]	
Viva	3	Able to answer the experiment related questions [3 Marks]	Partially answered the experiment related questions [1-2 Marks]	Not able to answer the experiment related questions [0 Marks]	
Total					

Ex. No. 11: ARRAYS IN PYTHON

Aim

To write a python program using input and output Functions

Theory

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
car1 = "Ford"
car2 = "Volvo"
car3 = "BMW"
```

An array can hold many values under a single name, and you can access the values by referringto an index number.

Access the Elements of an Array
The array elements can be accessed by index number.

Example
Get the value of the first array item:
```
x = cars[0]
```

Modify the value of the first array item:
```
cars[0] = "Toyota"
```

The Length of an Array
Use the len() method to return the length of an array (the number of elements in an array).

Example
Return the number of elements in the cars array:

```
x = len(cars)
```

Note: The length of an array is always one more than the highest array index.

Looping Array Elements
The for in loop used to loop through all the elements of an array.

Example
Print each item in the cars array:

```
for x in cars:
```

print(x)

Adding Array Elements
The append() method used to add an element to an array.

Example
Add one more element to the cars array:

```
cars.append("Honda")
```

Removing Array Elements
The pop() method used to remove an element from the array.

Example
Delete the second element of the cars array:

```
cars.pop(1)
```

The remove() method also used to remove an element from the array.Example
Delete the element that has the value "Volvo":

```
cars.remove("Volvo")
```

Note: The list's remove() method only removes the first occurrence of the specified value.

Array Methods
Python has a set of built-in methods that you can use on lists/arrays.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position pop()
	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Note: Python does not have built-in support for Arrays, but Python Lists can be used instead.

Problem Description:

Write a program to find the numbers which are not multiple of 5 from the user list. Print the numbers in a single line separated by a space.

Result:

Thus a python program to find the numbers which are not multiple of 5 from the user list is written and executed successfully.