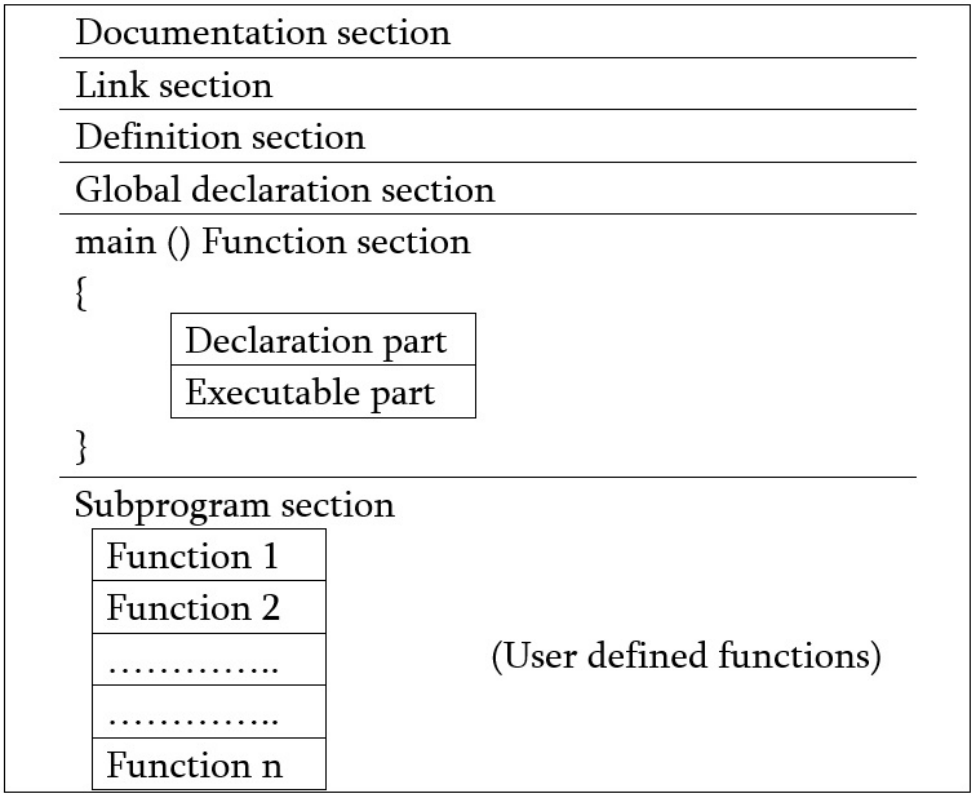
Solution/Model Answer
C Programming for Problem Solving 18CPS13
Jan 2019 exam
Dr. P. N. Singh, Professor(CSE)

---

1. Explain the basic structure of a C Program with example. **10 Marks**

Ans:



- **Documentation Section:** This section is used to write Problem, file name, developer, date etc in comment lines within /*….*/ or separate line comments may start with // . Compiler ignores this section. Documentation enhances the readability of a program.
- **Link section :** To include header and library files whose in-built functions are to be used. Linker also required these files to build a program executable. Files are included with directive # include
- **Definition section:** To define macros and symbolic constants by preprocessor directive #define
- **Global section:** to declare global variables – to be accessed by all functions
- **main() i**s the user defined function which is recognized by the compiler first. So, all C program must have user defined function main() { ……. }. It should have declaration part first then executable part.
- **Sub program section:** There may be other user defined functions to perform specific task when called.

```
/* Example: a program to find area of a circle – area.c
            Dr. P. N. Singh                - Documentation Section*/
#include <stdio.h>                        /* - Link/Header Section */
#define PI 3.14                        /* definition/global section*/
int main()                             /* main function section */
{
     float r, area;                   /* declaration part */
     printf("Enter radius of the circle : "); /* Execution part*/
     scanf("%f", &r);
     area=PI*r*r; /* using symbolic constant PI */
     printf("Area of circle = %0.3f square unit\n", area);
     return (0);
}
```

1.b. Define a variable. Explain the rules for constructing variables in C language. 4 Marks

Ans:

A variable is an identifier for which a value can be assigned and changed. It is name of the location of memory where some value is stored temporarily.

Rules to construct variable:

- ➔ They may contain A-Z, a-z, and _ (underscore) (no blanks)
- ➔ An identifier name but must be started with an alphabet or _ (specific purposes)
- ➔ Maximum length of name of identifier should be 31 characters (ANSI C standards) however in newer versions length of identifier name may be more.
- ➔ Keywords & compiler constants should not be identifier name

Valid identifiers: num1, a_2

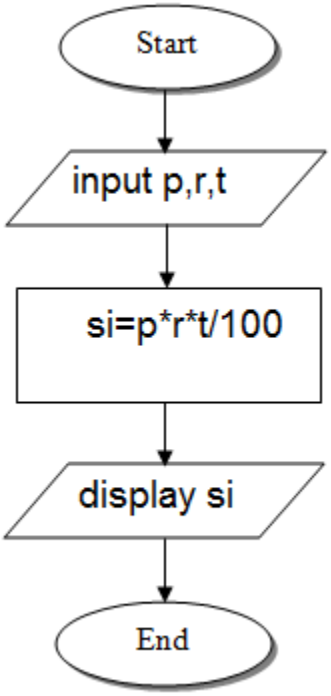
Invalid identifiers: $num1, for, 1st_paper_marks

1. c. Write a C Program to compute simple interest. Draw the flow chart for the same. 6 Marks

Ans:

```
/* Program to calculate simple interest */
#include <stdio.h>
int main( )
{
        float p,r,t,si;
        printf("Enter principle, rate and time : ");
        scanf("%f%f%f", &p, &r, &t);
        si=p*r*t/100.00;
        printf("Simple interest = %f\n", si);
        return (0);
}
```

Flowchart to compute Simple Interest



2 a. Define data types. Explain primitive data types supported by C language with example 10 Marks

Ans:

A data type is a classification of data, which can store a specific type of information.

Primitive data types are primary, fundamental or predefined types of data, which are supported by the programming language. In C primitive data types are:

- char
- int
- float
- double

Programmers can use these data types when creating variables in their programs. For example, a programmer may create a variable called marks and define it as a int data type type.

Examples:

```
int marks;
double num = 0.00;
char c;
float sum;
```

Derived data types are non-primitive data types or composed data types from primary data types. Non-primitive data types are not defined by the programming language, but are instead created by the programmer. Arrays, pointers, structures are examples of derived data type.

Examples of derived/non-primitive data types:

```
int a[10]; /* array of 10 integers */
struct student
{
        int roll;
        char name[20];
        int marks;
};
struct student s1,s2; /* s1 and s2 are structure variables or objects */
int *p; /* p is pointer variable to keep address of integer data */
```

2. b. List all operators used in C language and evaluate the following expression. 4 Marks

Ans:

List of operators used in C Language:

Ans:

i. Arithmetic Operators :
   + (addition)
   - (subtraction)
   * (multiplication)
   / (division)
   % (modulus)

ii. Relational Operators:
   < (less than)
   <= (less than or equal to)
   > (greater than)
   >= (greater than or equal to)
   ==(equal to)
   !=(not equal to)

iii. Logical Operators:
   && (and)

|| (or)
! (Not)

iv. Increment & Decrement:
++, --

v. Assignment Operator :
=, +=, -=, */, /=, %=

vi. Pre-processor Operator : #

vii. Member operator: . and ->

viii. Bitwise Operators : & (Bit-wise AND), | (Bit-wise OR), ! (Bit-wise NOT), ^ (Bit-wise XOR), >> (Bit-wise right shift), <<(Bit-wise left shift)

ix. Ternary Operator(Conditional if): ? :

x. Address of operator: &

xi. Pointer (dereference) operator: *

xii. Comma Operator: ,

xiii. Statement terminator operator: ;

Evaluation of expression:

i. x=a-b/3 + c*2 -1 when a = 9, b=12, c=3

Ans: 9-12/3 + 3*2 – 1 = 9-4 + 6 - 1 = 10 Ans

ii. 10 != 10 || 5 < 4 && 8

Ans. 0 /* (Logical then relational, in logical first not then and then or) */

2 c. Describe the various type of computer **6 Marks**

Ans:

According to purpose:

a. **Analog computers:** Analog computer is a form of computer that is used to read & measure the physical phenomena such as electrical, mechanical, or hydraulic quantities **(Not discrete)** . Eg. Thermometer, Speedometer, Morse code Telex Machine, amplifiers etc.
b. **Digital Computer:** A computer that performs calculations and logical operations with quantities represented as digits, usually in the binary number system. Example: Calculator, laptops, PCS
c. **Hybrid Computer:** A combination of analog and digital computers those are capable of inputting and outputting in both digital and analog signals. A hybrid computer system setup offers a cost effective method of performing complex simulations. **Here calculations are done digitally and actions are taken mechanically or vice-versa. Super computers and robots are hybrid computers.**

According to size:

a. Super Computer: They are very large and very fast & task specific computers used by large organizations. These computers are used for **weather forecasting, meteorology, earth-quack studies, research and exploration** purposes. Super computers may be installed in satellites. NASA uses supercomputers for launching space shuttles, controlling them and for space exploration purpose. Example: **Titan, Cray(of India)**
b. Mainframe Computer: These are smaller and less faster than super computers. Large firms & government organizations like banks, railways use Mainframes to run their business operations in their head/central office as a central role to update day to day transactions. Using virtual machines, mainframes run the various operating systems as if they were running on different computers. Example z9, z10
c. Mini Computers: Minicomputers are used by small businesses & firms. Minicomputers are also called as “Midrange Computers”. Individual departments of a large company or organizations use Mini-computers for specific purposes. Example K202, IBM Midrange
d. Micro Computers: Desktop computers, **laptops, PCs, personal digital assistant (PDA), tablets & Smartphones, Scientific Clculators, Notebook, Gaming console, Sound and navigation System** are all types of

microcomputers. The micro-computers are widely used & the fastest growing computers. Well known manufacturers are Dell, Apple, Samsung, Sony etc.

3 a. Explain the formatted I/O functions of C Language with syntax and example. 4 Marks

Ans:

There are 2 imported formatted I/O functions of C Language printf() & scanf()
Again its is extended in file handling function with pointers as fprintf() and fscanf()

printf() – Library function for formatted output:

Output data can be written on to a standard output device using the library function printf(). The printf statement provides certain features that can be effectively exploited to control the alignment and spacing of printouts on the terminals. The general form of printf statement is:

printf ("control string", arg1,arg2, ..., argn);

Control string of printf function consists of three types of item :

- Character that will be printed on the screen as they appear.
- Format specification that define the output format for display of each item.
- Escape sequence characters such as \n,\t, and \b.

scanf() – Library function for formatted input

Formatted input refers to an input data that has been arranged in a particular format. Input data can be entered into the computer from a standard input device by means of the C library function scanf. In general terms, scanf function is written as

scanf ("control string", arg1, arg2, ..., argn);

The control string specifies the field format in which the data is to be entered and the arguments arg1,arg2.....,argn specify the address of locations where the data is stored. Actually arguments are pointers which indicate where the data items are stored in computer's memory.

Control string contains field specification which direct the interpretation of input data. It may include:

- Field (or format) specification, consisting of the conversion character %, a data type character (or type specifier, and an optional number, specifying the field width.
- Blanks, tabs, or new lines.

- The arguments are written as variables or arrays, whose type match the corresponding character groups in the control string. Each variable name must be preceded by an ampersand (&). Array names should not begin with an ampersand.

3. b. Write a C Program to implement commercial calculator using switch statement. 4 Marks

Ans:

```
/* calculator by switch */
int main( )
{

        float n1,n2,result;
        int choice;
        printf("Enter 2 numbers : ");
        scanf("%f%f", &n1, &n2);
        printf("1. Addition 2. Subtraction 3. Multiplication 4. Division:\n");
        printf("Enter your choice number : ");
        scanf("%d", &choice);
        switch(choice)
        {
        case 1 : result = n1+n2;
```

```
            break;
        case 2 : result = n1-n2;
            break;
        case 3 : result = n1*n2;
            break;
        case 4 : if(n2 == 0)
                { printf("divide error!!!\n"); return (99); }
            else
                result = n1/n2;
            break;
        default: printf("Wrong choice!\n");
        }
        printf("Result = %f\n", result);
        return (0);
}
```

Expected Output:
Enter 2 numbers : 50 25
1. Addition 2. Subtraction 3. Multiplication 4. Division:
Enter your choice number : 4
Result = 2.000000

3. c. Write the syntax of different branching statements and explain their working **10 Marks**

Ans:

In C programming language branching statements are:

if, if else & switch

if/if else can be extended to:
ladder if & nested if

if statement: syntax

```
if (expression/condition)
{
    statement1;
    statement2;
    ...
}
```

Example:

```
#include <stdio.h>
int main()
{
   int age;
   printf("Enter your age : ");
   scanf("%d", &age);
   if(age >= 25)
      printf("Celebrate valentine day\n");
   else
      printf("Just keep looking cute only.\n");
   return (0);
}
```

Nested if syntax:
if (expression/condition)

```
{
        if (expression/condition)
        {
                statement1;
                statement2;
                .....
        }
        else
        {
                statement1;
                statement2;
                .....
        }
}
else
{
        if (expression/condition)
        {
                statement1;
                statement2;
                .....
        }
        else
        {
                statement1;
                statement2;
                .....
        }
}
```

```
// Example: to find maximum in 3 numbers
#include <stdio.h>
int main()
{
    int n1,n2,n3,max;
    printf("Enter 3 numbers : ");
    scanf("%d%d%d", &n1, &n2, &n3);
    if(n1>n2)
        if(n1>n3)
            max=n1;
        else
            max=n3;
    else
        if(n2>n3)
            max=n2;
        else
            max=n3;

    print("Maximum = %d\n", max);
    return (0);
}
```

switch statement:

C has a built-in multiway decision statement known as a switch. It tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated with the case is 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, transferring the control to the next statement following the switch (where branch ends with }). 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.

switch statement syntax:

```
switch (expression)
{       case  condition1
                statement1;
                statement2;
                ......
                break;
        case condition2
                statement1;
                statement2;
                .......
                break;
        .....
        default:
                statement1;
                statement2;
                .......
}
```

```
/* Example of switch */
#include <stdio.h>
int main()
{
   int salary,bonus;
   char grade:
   printf("Enter grade : ");
   scanf("%c", &grade);
   printf("Enter salary : ");
   scanf("%d", &salary);
   switch (grade)
   {
      case 'a':
      case 'A':   bonus=salary;
              break;
      case 'b':
      case 'B':   bonus=salary+5000;
              break;
   default :  bonus=salary+10000; /*lower grade-more bonus*/
   }
   print("Bonus = %d\n", bonus);
   return (0);
}
```

4. a. Differentiate between while loop and do-while loop. Explain with syntax and examples. 8 Marks

Ans:

- ***while*** is a pre-tested (entry-controlled) loop. Condition is checked before entering in the loop.
- ***do-while*** is a post-tested (exit-controlled) loop. Condition is checked after execution of the statements within the loop.

- **The main difference is that do-while loop must execute once (because condition is checked at the end.)**

while statement syntax:

```
while (condition)
{
    statement1;
    statement2;
    ……
}
```

```
/*working example – to display squares of numbers from 1 to 20 */
#include <stdio.h>
int main()
{
    int x;
    x=1;
    while(x<=20)
    {
        printf("%d  %d\n", x, x*x);
        x++;
    }
    return (0);
}
```

do…while statement syntax: (It is a post tested loop & must executes at least once)

```
do
{
    statement1;
    statement2;
    ……
} while (condition);
```

```
 /*working example – to display squares of numbers from 1 to 20 */
#include <stdio.h>
int main()
{
    int x;
    x=1;
    do
    {
        printf("%d  %d\n", x, x*x);
        x++;
    }while(x<=20);
    return (0);
}
```

4. b. Write a program to fins sum of N natural numbers using for loop **4 Marks**

Ans:

```
/* Sum of N Natural numbers */
#include <stdio.h>
int main()
{
    int N, x, sumN=0;
    printf("Enter range ( value of N) to find sum of N natural numbers : ");
```

```
        scanf("%d",&N);
        for(x=1;x<=N;x++)
                sum+=x;
        print("Sum of %d natural numbers = %d\n",N,sum);
        return (0);
}
```

4. c. Write a C program to plot Pascal's triangle. **8 marks**

Ans:

```
/* Pascal's triangle - Dr. P. N. Singh */
#include <stdio.h>
int main()
{
  int x,y,z,p;
  for(x=0;x<9;x++)
  {
     for(y=1;y<20-x;y++)
         printf("  "); /* for blank spaces */
     for(y=0;y<x;y++)
     {
          if(x==0 || y== 0)
               p=1;
          else
               p=p*(x-y)/y;       /* For Bionomial co-efficient */

          printf("%4d",p);      /* printing value taking 4 minimum spaces */

     }
     printf("\n");
  }
  return (0);
}
```

Expected output: Plotting Pascal's 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
      1   7  21  35  35  21   7   1
```

5. a. Define array. Write the syntax for and with declaring and initializing 1D and 2D array with suitable example. 10 marks

Ans:

An array is an identifier that refers to the collection of data items which all have the same name. The individual data items are represented by their corresponding array elements. Address of an element (subscript) ranges from 0 to n-1, where n is total number of elements.

i) Declaration of one-dimensional array:

One dimensional array:

syntax:

type variable[size];

Examples:

```
float height[50];
int batch[10];
char name[20];
```

ii) **Initialization of one-dimensional array:**

Examples:

```
int marks[5] = {76,45,67,87,92};
static int count[ ] = {1,2,3,4,5};
static char name[5] = {'S', 'I', 'N', 'G', 'H', '\0'};
```

iii) Declaration of Two-dimensional array:

syntax:

type *arrayname*[rowsize][columnsize];

Examples:

```
int stud[16][5];
```

iv) Initialization of Two-dimensional array:

Examples:

```
int marks[3][3] = {
                    {26,57,66},
                    {56,77,48},
                    {76,54,82}
                  };

int marks[3][3] = { {12},{0},{0}};
```

Here first element of each row is explicitly initialised to 12 while other elements are automatically initialized to zero.

5. b. Write a C program to find the transpose of a matrix. **10 Marks**

Ans:

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

#define m 5
#define n 4
main()
{
  int m1[m][n],r,c,temp;
  printf("Enter %d x %d matrix:\n",m,n);
  for(r=0;r<m;r++)
   for(c=0;c<n;c++)
     scanf("%d",&m1[r][c]);

   /* now transposing */
   for(r=0;r<m;r++)
     for(c=r+1;c<n;c++) /* y=x+1 is the logic */
     {
       temp=m1[r][c];
       m1[r][c]=m1[c][r];
       m1[c][r]=temp;
     }

   printf("The  Transposed  Matrix\n");
   for(r=0;r<n;r++)
```

```
 {
  for(c=0;c<m;c++)
    printf("%5d",m1[r][c]);

  printf("\n");
 }
return 0;
}
```

Expected Output:

Enter 5 x 4 matrix:
3 2 4 5
2 4 3 6
7 6 5 4
4 5 6 7
2 1 5 4
The Transposed Matrix

3	2	7	4	2
2	4	6	5	4
4	3	5	6	5
5	6	4	7	2

6. a. Define string. List out all string manipulating function. Explain any two with example 10 Marks
Ans:

String manipulating Functions: (Header file: string.h)

String manipulating Function	Work of Function
strlen()	Calculates the length of string
strcpy()	Copies a string to another string
strcat()	Concatenates(joins) two strings
strcmp()	Compares two string
strrev()	Reverses a string
strlwr()	Converts string to lowercase
strupr()	Converts string to uppercase
strchr()	Scans a string to search a character

Now explaining strcat() and strcmp():

strcat(str1, str2) – Concatenates(appends) str2 to str1 at the end.
Syntax:

```
char* strcat (char* strg1, const char* strg2);
```

Example:
If two strings are:

```
char name[]="Roma";
```

char surname[]=”Ritambhara”;

Then to append “Ritambhara” with name “Roma”

strcat(name,surname);

printf(“%s”, name);

Output: Roma Ritambhara

strcmp(str1, str2) – Compares two string and returns:

- **0 if both strings are identical**
- **else returns difference based on ASCII value of first mismatch.**

Syntax:

int strcmp (const char* str1, const char* str2);

Example:

printf(“%d”, strcmp(“SINGH”, “SINHA”);

output: - 1

6. b. Write a C Program for [considering integer data] i) Bubble sort ii) Linear serach marks 10

Ans:

```
/* Code Bubble Sort */
#include <stdio.h>
void selesort(int a[], int size)
{
  int x,y,temp;
  for(x=0;x<size-1;x++)
   for(y=0;y<size-x-1;y++)
    if(a[y] > a[y+1])
     {                    /* swapping adjacent element */
          temp=a[y];
          a[y]=a[y+1];
          a[y+1]=temp;
     }
}

main()
{
 int a[1000], tot, x;
 printf("Total elements : ");
 scanf("%d",&tot);
 for(x=0;x<tot;x++)
 {
          printf("Enter element %d :", x+1);
          scanf("%d",&a[x]);
 }
 selesort(a,tot);
 printf("Sorted array:\n");
 for(x=0;x<tot;x++)
          printf("%5d",a[x]);

return (0);
}

/* II. Linear search */

int linser(int a[], int size, int skey)
{
```

```
 int x,found=0;
 for(x=0;x<size;x++)
  if(a[x]==skey)  /* searching sequentially */
   {
          found=1;
          break;
   }
 if(found) return (x+1);
 else return (NULL);
}


int main()
{
 int a[1000],x,k,tot;
 printf("How many elements? : ");
 scanf("%d",&tot);
 for(x=0;x<tot;x++)
 {
          printf("Enter element %d : ",x+1);
          scanf("%d",&a[x]);
 }
 printf("Enter element to search (by recursive function): ");
 scanf("%d",&k);
 x=lsrec(a,tot,k);  /* calling function */
 if(x)
  printf("Found at %dth position.\n",x);
 else
  puts("Not found");
return (0);
 }
```

Q 7 a. What is a function? Explain different type of functions based on parameter. 10 marks

Ans:

- A function is a self-contained block of statements that performs a coherent task of some kind.
- Mathematical definition of a function is like that "A function f(x) returns value for the argument x."
- Functions are classified of two types – In-built function and User/Programmer defined functions.
- A function name is followed by a pair of ( ).
- A function may or may not have arguments. Arguments are written within parentheses separating by , (comma).

Yes, main( ) is a user defined function which is specially recognised function in C. Every program must have a function main( ) to indicate where the program has to begin its execution.

Function prototyping : Formal declaration of a function with data type like variables, arrays prior to its use is known as function prototyping. Without it most of the compilers displays an error message "abcd function should have a prototype." Called function should receive the same data type as declared.

Function based on argument:

A function may or may not send back any value to the calling function. The data type void says that the function will not return any value and suppresses the warning message that "Function should return a value." If any function returns any value, it is done through the return statement.

return; or return (expression);

Function without parameter:

```
void gao()
{
   printf("Kudi Punjaban dil chura ke lay gai – Sona Sona, Dil Mera Sona);
}
int main()
{
      gao();  /* calling function */
      return (0);
}
```

A function may have arguments to pass. Formal declaration of the data type of arguments to be passed with function should be done also.

```
int area( int l, int w) { return (l*w);
int main()
{
    int len, wid;
   printf("Enter length and width of rectangle : ");
  scanf("%d%d",&len,&wid);
  printf("Area of rectangle = %d square unit\n", area(len,wid));
 return (0);
}
```

In the called function arguments are received as parameters declaring their type:

```
                int mul( int a, int b)
                {
                        ......
                }
or              int mul(a,b)
                int a,b;   /* T & R classical style */
                {  ......  }
```

A function may be nested also:

```
int max2(int a, int b) { return (a>b?a:b); }
int main( )
{
     int x,y,z;
     printf("Enter 3 numbers : ");
     scanf("%d%d%d",&x,&y,&z);
     printf("Maximum = %d\n". max2(max2(x,y),z));  /* Nesting of function if more arguments */
     return (0);
}
```

7. b. Write a program to find factorial of a given number using function. 4 Marks

Ans:

```
/* Program to find factorial of using recursion */
#include <stdio.h>
long int fact(int m)
{
if (m == 1 || m == 0)
           return(1);
else
      return(m * fact(m-1));      /* recursion */
}
```

```
int main()
{
      int n;
      printf("Enter any number for factorial : ");
      scanf("%d",&n);
      printf("Factorial of % d = %ld\n",n,fact(n));
      return (0);
}
```

O/P:

```
Enter any number for factorial : 6
Factorial of 6 = 720
```

7. c. Write a program to find GCD & LCM of two numbers using concept of function

Ans:

```
/* GCD & LCM */
#include <stdio.h>
/* GCD using recursive function*/
int GCD(int dividend,int divider)
{
  int rem=dividend%divider;
  if(rem==0)
   return (divider);
  else
   return GCD(divider, rem);
}

/* LCM of 2 numbers using simple function*/
int LCM(int a, int b)
{
   int lcm,x;
   for(x=b;a<=a*b;x++)
   if(x%a == 0 && x%b ==0 )
   {
      lcm = x;
      break;
   }
   return (lcm);
 }


main()
{
   int n1,n2;
   printf("Enter two numbers : ");
   scanf("%d%d",&n1,&n2);
   printf("GCD = %d\n",GCD(n1,n2));
   printf("LCM = %d\n",LCM(n1,n2));
   return (0);
}
```

8. a. Explain recursion and write a program to find nth term of Fibonacci series. **10 marks**

Ans:

When a function calls itself repeatedly, is called recursion.

- An useful example of recursion is to find factorial value of any number.
  Factorial of n:
  n! = n*(n-1)*(n-2)*(n-3)*.....1

or
n! = n*(n-1)!

- An other example of recursion can be used also in Fibonacci series.
  Fibonacci was a mathematician. He got a problem. A pair of rabbit can produce a pair of rabbit in some constant duration. The new born pair becomes fertile in the same period. He had to calculate total pairs after number of that constant periods. He generated a series of numbers where current number is sum of last two numbers as:

  0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,…

```
/* Fibonacci series up to nth terms using recirsion */
#include <stdio.h>

int fibo(int n)
{
  if (n==1)
    return (0);
  if (n==2)
    return (1);
  else
    return (fibo(n-1)+fibo(n-2));
}

int main()
{
  int n,x;
  printf("Enter nth term for Fibonacci series : ");
  scanf("%d",&n);
  for(x=1;x<=n;x++)
     printf(" %d",fibo(x));

  return (0);
}
```

Output:
Enter nth term for Fibonacci series : 10
0 1 1 2 3 5 8 13 21 34
Enter nth term for Fibonacci series : 20
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

8. b. Give the scope and life time of following:
i. External variable ii. Static variable iii. Automatic variable iv. Register variable
Ans:

The storage class specifies the portion of the program within which the variables are recognized. variables can be categorized by storage class as well as data type. Storage class refers to the permanence of a variable and its scope within the program, that is, the portion of the program over which the variable is recognized.

i. extern (External/Global variables): They are global variables, known to all the functions of the program from point of definition. Hence, they usually span more than one functions, and often an entire program. Any of the function can change/alter/corrupt extern variables. Variables defined outside of the program are by default of extern storage class.

```
extern int global = 200; /* by default extern */
   void func()
   {
            printf("global = %d\n",global); /* displays global = 300 */
            global=400;  /* global variable changed by func() */
   }
```

```
int main( )
    {
        printf("global = %d\n",global);
        global=300; /* value of global changed */
        func(); /* calling function */
        printf("global = %d\n",global); /* displays global = 400 */
        return (0);
}
```

ii. static (Static variable): Static variables have the same scope as automatic variables, i.e. they are local to the functions in which they are defined. Unlike automatic variables, however, static variables retain their values throughout the life of the program. As a result, if a function is called again, the static variables defined in the called function will retain their former values **(They will not be reinitialized again. They are confined at the first initialization)**.

```
#include <stdio.h>
int func()
{
  static int k = 0; /* by default also zero */
  k++;
  return (k);
}

int main()
{
   int x, sum=0;
  for(x=1;x<=5;x++)
    sum+=func();
 print("sum = %d\n", sum);
 return (0);
}
```

iii. auto (Automatic variables): They are local variables, available to the function in which it is declared. Their scope is confined to that function only. Value of the variable defined within a function is no way available to other function. Variables declared within a function are by default of auto storage class.

```
int man()
{
    auto int x,y,z;
    int num;        /* num is also interpreted as auto storage class
        /* ….. other code */
}
```

iv. register (Register variables) : Registers are special storage area within a computer's central processing unit (In-built memory with CPU). The actual arithmetic and logical operations that comprise a program are carried out within this registers. The execution time can be reduced considerably if certain values can be stored within these registers rather than in computer's memory.

The register variables are local to the function in which they are declared. The address of operator cannot '&' cannot be applied to register variables.

```
register  int  x, y, z;
```

9. a. What is a structure? Explain the syntax of structure declaration in C with example. 4 Marks

Ans:

Structure is a data structure whose individual elements can differ in type. It may contain integer elements, character elements, pointers, arrays and even other structures can also be included as elements within a structure. struct is keyword to define a structure.

```
struct  tag  {
        type member1;
            type member2;
            type member3;
        ......
        type member n;
    };
```

New structure type variables can be declared as follows:
struct tag var1, var2, var3, ……. varn, var[10];
here var[10] is array of structure variable (object)

9. b. Write note on: i) Arrays within structure ii) arrays of structure 4 marks

Ans:

i) Array of structure:

Whole structure can be an element of the array. A student structure with members roll, name and marks:

```
struct student
{
        int roll;
        char name[30];
        int marks;
};
```

Now we can use this template to create array of 60 students for their individual roll, name and marks.

struct student s[60]; /* array of structure *

to access roll of student x : **s[x].roll**

ii) Array within structure:

A member of structure may be array. In above example name is also array of characters. We can create a structure of student with members roll, name & marks (array of integers) for 6 papers:

```
struct student
{
        int roll;
        char name[30];
        int marks[6]; /* array within structure */
}s[60];
```

To access the marks of student x in paper y: **s[x].marks[y]**

9. c. Implement structures to read, write and compute average of marks and the students scoring above and below average marks for class of N students.

Ans:

```
/* computing average of marks in students' structure */
#include <stdio.h>
#include <stdlib.h>
struct student
{
   int roll;
   char name[30];
   int marks;
};
struct student *s;   /* pointer to structure variable object */

int main()
{
```

```
  int N,x,sum;
  float avg;
  printf("Enter total students : ");
  scanf("%d",&N);
  s=malloc(sizeof(N));   /* allocating memory for N students */
  sum=0;
  for(x=0;x<N;x++)
  {
     printf("Enter roll,name & marks for student %d : ", x+1);
     scanf("%d%s%d",&s[x].roll,s[x].name,&s[x].marks);
     sum+=s[x].marks;
  }
  avg=(float)sum/N;
  printf("Average of marks = %.2f\n",avg);
  printf("List of student scoring above average:\n");
  for(x=0;x<N;x++)
    if(s[x].marks >= avg)  /* above average */
       printf("%5d%30s%5d\n",s[x].roll,s[x].name,s[x].marks);

  printf("List of student scoring below average:\n");
  for(x=0;x<N;x++)
    if(s[x].marks < avg)     /* below average */
       printf("%5d%30s%5d\n",s[x].roll,s[x].name,s[x].marks);
  free(s);
  return (0);
 }
```

Expected Output:

Enter total students : 5
Enter roll,name & marks for student 1 : 111 Shreya 76
Enter roll,name & marks for student 2 : 222 Ashutosh 79
Enter roll,name & marks for student 3 : 333 Raghu 54
Enter roll,name & marks for student 4 : 444 Kumar 65
Enter roll,name & marks for student 5 : 555 Santosh 60
Average of marks = 66.80
List of student scoring above average:

111	**Shreya**	**76**
222	**Ashutosh**	**79**

List of student scoring below average:

333	**Raghu**	**54**
444	**Kumar**	**65**
555	**Santosh**	**60**

10. a. What is a pointer? Show how pointer variable is declared and initialized. 5 marks

Ans:

"Pointer is a variable that stores address(location) of data as value."
In C when we define a pointer variable we do so by preceding its name with an asterisk.

To declare:
datatype *pointervar;
int *p;

```
/*Example code */
#include <stdio.h>
int main()
{
```

```
        int n=20,*p;
        p=&n; /* address of n is initialized to pointer variable p */
        /*Now value of n can be accessed by variable or by address  */
        /*p value is value and p is address */
        printf("Value of n = %d\n", n);
        printf("Address of n = %u\n", p);
        printf("Value stored at address = %d\n", *p);
        *p=40;
        printf("Value changed by pointer = %d\n", n);  /* value of n is changed to 40 */
        return (0);
}
```

10. c Write a C program to find sum and mean of all array elements in an array using pointer. 10 marks

Ans:

```
/* sum & mean */
#include <stdio.h>
#include <stdlib.h>
int main()
{
        double *a,sum=0.00,mean;
        int x,n;
        printf("How many elements : ");
        scanf("%d",&n);
        a=malloc(sizeof(double)*n);  /* allocating memory */
        for(x=0;x<n;x++)
        {
                printf("Enter number %d : ", x+1);
                scanf("%lf",(a+x));
                sum+= *(a+x);      /* using pointers */
        }
        mean=(double)sum/n;
        printf("Sum=%.3lf Mean=%.3lf \n" , sum,mean);
        free(a);
        return (0);
}
```

Expected Output:
How many elements: 5
Enter number 1: 10
Enter number 2: 20
Enter number 3: 30
Enter number 4: 40
Enter number 5: 50
Sum=150.000 Mean = 30.000