# Model Answer
# VTU-PSP-21PSP13-May 2022

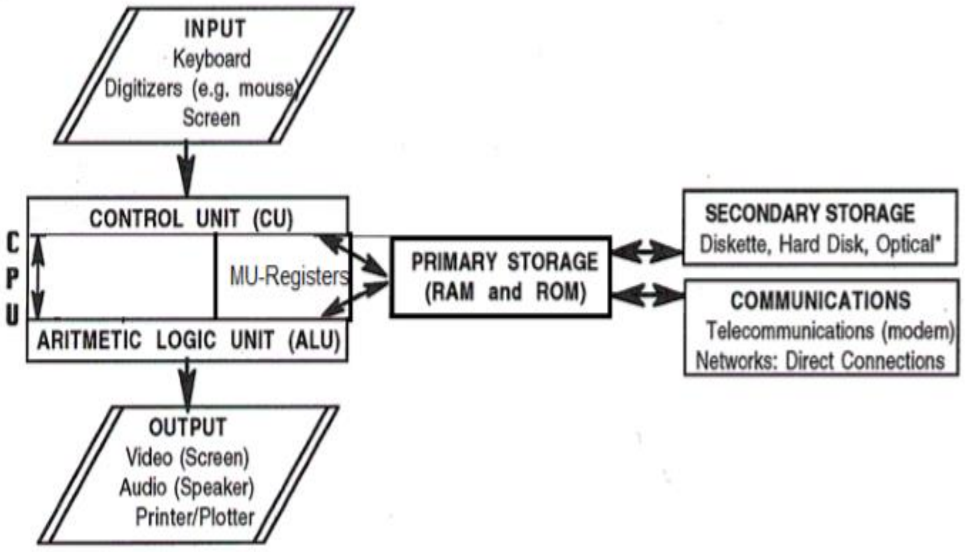
Dr. Paras Nath Singh, Professor(CSE/ISE)

1.

**a. Explain components of a computer with block diagram 10 M**

**Ans:**

Block diagram of Computer components

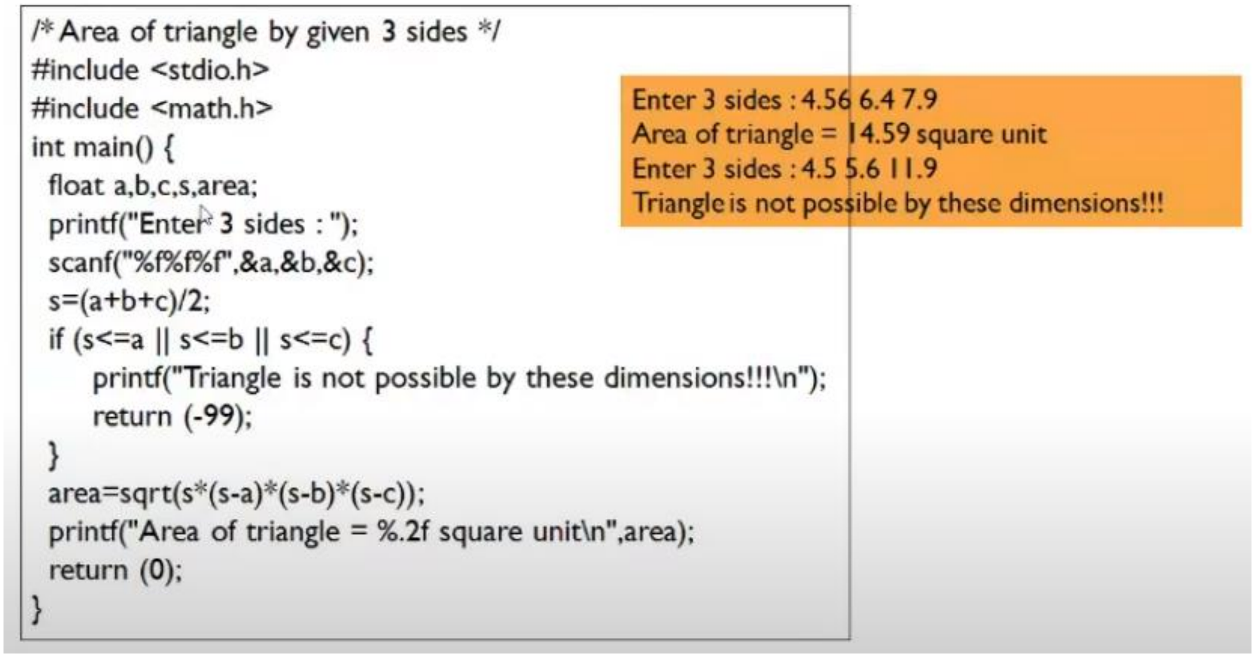


- **Input unit:** The input unit that links the external environment to input data & tasks with the computer system to execute. Keyboard is used for characters input. Mouse is used in GUI (Graphic User Interface). Internally data is processed in machine readable form.
- **Output Unit:** Output/result is displayed, printed & transmitted to outside world. There are many output devices: monitor, printer/plotters, display boards, speaker etc.
- **Storage unit:** The data and instructions that are entered into the computer system through input units have to be stored inside the computer before the actual processing starts. The storage unit is Primary Memory (RAM) & Secondary (permanent storage devices: disks, tapes)
- CPU (Central processing Unit): It is the main unit which controls all events within computer. The CPU has 3 internal units: CU, ALU & Registers:
  - **CU(Control unit):** The control unit acts as a central nervous system for the other components of the computer. It manages and coordinates the entire computer system. It obtains instructions from the program stored in main memory, interprets the instructions, and issues signals that cause other units of the system to execute them.
  - **ALU (Arithmatic & Logic Unit):** The arithmetic and logic unit (ALU) is the part where actual computations take place.
  - **MU (Memory Unit/Registers):** Registers are built-in memory with CPU having less storage space in bits. Instruction Registers, Address registers, Program Counters, Accumulators are example of registers. ALU takes data from here inside the CPU.

- **RAM(Random Access Memory):** RAM is the memory - primary storage where our data & programs are stored temporarily. It is volatile in nature. After switching off the system everything will be vanished from RAM.
- **ROM(Read Only Memory):** ROM is storage medium/"firmware" where some code of manufacturer is permanently hardwired in chip which always executes automatically when we start the system. The process is known as POST(Power on Self test). Booting preceeds POST.

**b. Write a C Program to find area of a triangle for the given 3 sides and draw the flow chart.**
**4 M**

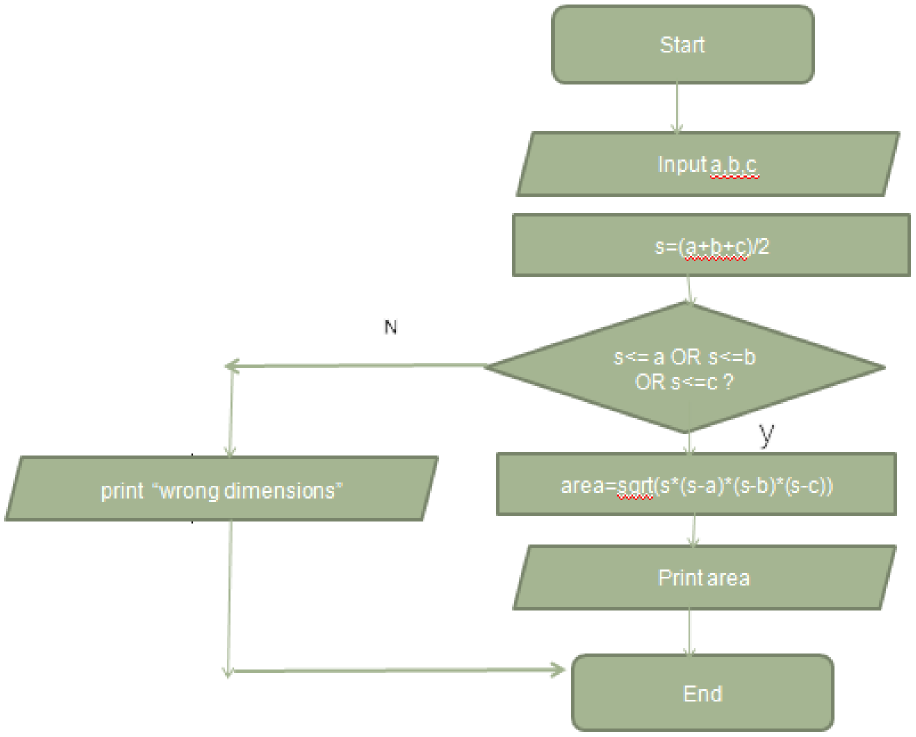
Ans:



```
/* Area of triangle by given 3 sides */
#include <stdio.h>
#include <math.h>
int main() {
  float a,b,c,s,area;
  printf("Enter 3 sides : ");
  scanf("%f%f%f",&a,&b,&c);
  s=(a+b+c)/2;
  if (s<=a || s<=b || s<=c) {
     printf("Triangle is not possible by these dimensions!!!\n");
     return (-99);
  }
  area=sqrt(s*(s-a)*(s-b)*(s-c));
  printf("Area of triangle = %.2f square unit\n",area);
  return (0);
}
```

```
Enter 3 sides : 4.56 6.4 7.9
Area of triangle = 14.59 square unit
Enter 3 sides : 4.5 5.6 11.9
Triangle is not possible by these dimensions!!!
```

Flow chart



c. **Explain various types of computers. 6 M**

Ans:

**Types of computer:**

- **According to purpose: Analog, Digital, Hybrid**
- **According to size: Super, Main Frame, Mini, Micro( PC-Desktop, Laptop, Palmtop, Note book, Tablet, Pocket- smart Phone)**

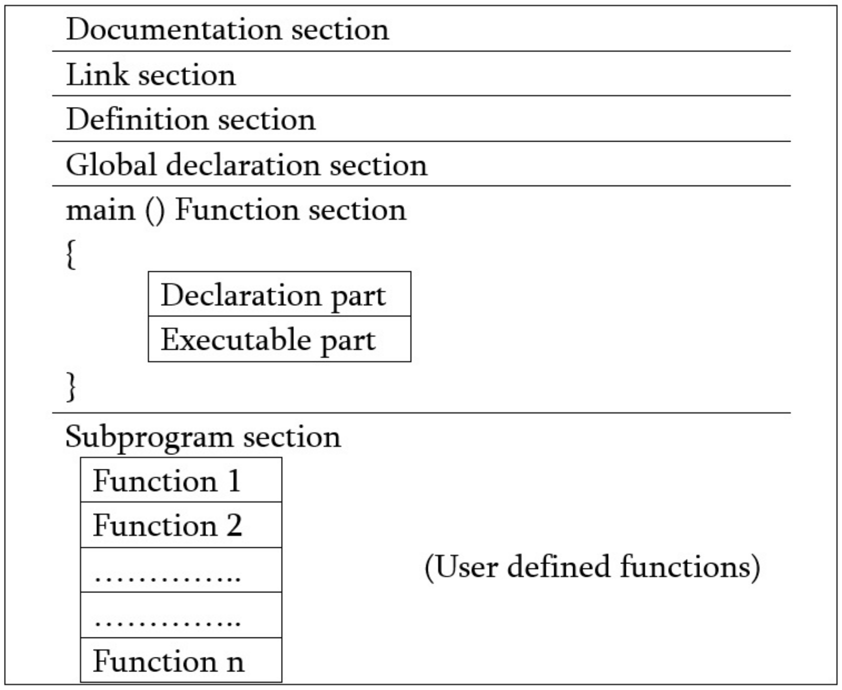
| Sr.No. | Type | Specifications |
|---|---|---|
| 1 | PC (Personal Computer) or Micro-Computers | It is a single user computer system having a moderately powerful microprocessor. It is termed as a computer that is equipped microprocessor as its CPU. |
| 2 | Workstation | It is also a single user computer system, similar to the personal computer, however, has a more powerful microprocessor. |
| 3 | Mini-Computer | It is a multi-user computer system, capable of supporting hundreds of users simultaneously. |
| 4 | Main Frame | It is a multi-user computer system, capable of supporting hundreds of users simultaneously. Software technology is different from minicomputer. |
| 5 | Super-Computer | It is an extremely fast computer, which can execute hundreds of millions of instructions per second. |

2.

a. **Write basic structure of a C program and give brief explanation for each sections with examples. 10M**

Ans:

Basic structure of a C Program



- **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
                    - 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);
}
```

**b. Define operator. Explain any 6 operators with examples. 7M**

**Ans:**

Operator is a symbol that performs operations on operands (value or variables). There are many types operators like arithmetic, logical, relational etc. Precedence of operators are according to their rank. If same rank operators are used the their associativity is checked (left to right or right to left).

**List of 6 operators used in C Language:**

Ans:

**i. Arithmetic Operators :**

+ (addition)
- (subtraction)
* (multiplication)

/ (division)
% (modulus)

Example:
int x=20, y=30;
printf("%d", x*y); /* the output will be 600 */

**ii. Relational Operators: (Booleans returns 1 for true and 0 for false)**

< (less than)
<= (less than or equal to)
> (greater than)
>= (greater than or equal to)
==(equal to)
!=(not equal to)

Example: printf("%d", x>y); /The output will be 0 because x is 20 and y is 30 */

iii. Logical Operators:

&& (and – both of the conditions must be satisfied for tru)
|| (or – either of the condition should be satisfied for true)
! (Not – negation of true is false and vice-versa)

Example: printf("%d", x >y || x <y ); /The output will be 1 one condition is satisfied */

iv. Increment & Decrement:

++ (the variable will be incremented by 1)
-- (the value will be decremented by 1

Example:
x++; or ++x; (is equivalent to x = x+1; )
y--; or –y; (is equivalent to y = y-1; )

**v. Assignment and augmented assignment Operators :**

=, +=, -=, */, /=, %=

**Right side value is assigned to left side variable**

Example:
X=50; /* 50 is assigned to variable x */

**vi. Pre-processor Operator : #**

This operator is as pre-compilation process of header files, library files and to define macros.
#include <stdio.h>
#define PI 3.14
include and define keywords are used with pre-processor operator # to become pre-processor directive.

**c. Check the following identifiers are valid or invalid.**
**i) sum100 ii) sum+3 iii) int iv) abcd v) X Y vi) 2product 3M**

Ans:
Name of valid identifier may contain alphabets (a-z, A-Z), numbers (0-9) and underscore _. It must NOT be started with a number. Keywords shouldnot be name of identifiers. Based on that:

| | | |
|---|---|---|
| i) | sum100 | Valid |
| ii) | sum+3 | Invalid (+ character is not allowed in identifier) |

| | | |
|---|---|---|
| iii) | int | Invalid (int is a keyword) |
| iv) | abcd | Valid |
| v) | X Y | Invalid (if space is between X and Y) |
| vi) | 2product | Invalid (Identifier name must not be started with a number |

3.

a. **Write syntax of different branching statements and explain with example how they work.** **10M**

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);
}
```

**b. Write a C program to perform all arithmetic operations for the given two integers using switch statements.** **6M**

**Ans:**

```
/* switch statement Example: Arithmetic operation 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 5. Modulus:\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;
            case 5 : 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
```

**c. With the help of example and syntax, explain input/output functions of C language. 4M**

Ans:

scanf() – Library function for formatted input

Formatted input refers to an input data that has been arranged in a particular format. C library function scanf () is used to read/scan data from standard input device (keyboard). In general terms, scanf function is written as

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

ExampeL: scanf("%d%s%d", &roll, name, &marks);

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. 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 preceded by & address of operator except arrays. array name itself represents address of first character.

**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 to control the alignment and spacing of printouts on the terminals. The general form of printf statement is:

```
printf (“control string”, arg1,arg2, ........., argn);
```

Example: prinft(“%d\t%s\t%d\n”, roll, name, marks);
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 can be in format specification.

**Other input functions:**
getchar( ) : to read a character from keyboard
gets() : to read a string from keyboard terminated by new line character
Other output functions:
putchar( ) : to print a character to screen (standard output)
pets() : to print a string to screen (standard output) terminated by new line character

4.

**a. Distinguish between while and do while statements. Explain with syntax and example 10M**

Ans:

- The while statement is entry controlled loop or pre-tested loop runs with testing the condition
- The do…while loop is an exit controlled loop or post-tested loop that must execute at least once irrespective of condition.

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);
}
```

**b. Wrte a C program to check whether given number is prime or not.. 6M**

Ans:

```
#include<stdio.h>
int main()
{
        int n,d,prime=1;
        printf("Enter a number : ");
        scanf("%d", &n);
        for (d=2;d<n/2;d++)
        {
                if (n%d==0)
                prime=0;
                break; /* there is no need to check further */
        }
        if (prime)
```

```
                printf("Yes %d is a prime number.\n", n);
        else
                printf("No, %d is not a prime number.\n", n);

        return (0);
}
```

**Expected output:**

```
Enter a number: 11
Yes 11 is a prime number
```

c. **Explain the use of break and continue inside-for loop with example**

Ans:
*break* statement would only exit from the loop containing it.

```
//Example program to check a number whether it is prime
#include<stdio.h>
int main()
{

        int n,d,prime=1;
        printf("Enter a number : ");
        scanf("%d", &n);
        for (d=2;d<n/2;d++)
        {
                if (n%d==0)
                prime=0;
                break; /* there is no need to check further */
        }
        if (prime)
                printf("Yes %d is a prime number.\n", n);
        else
                printf("No, %d is not a prime number.\n", n);

        return (0);
}
```

**The continue statement is used in loops to skip the following statements in the loop and to continue with the next iteration (current iteration in for loop).**

```
//Example program to enter marks (0-100) in 6 subjects for sum
#include<stdio.h>
int main()
{
        int marks,sum=0, x;
        for (x=1;x<=6;x++)
        {
                printf("Enter marks %d : ", x)
                scanf("%d", &marks);
                if(marks <0 || marks > 100)
                {
                        printf("Invalid marks!!!\n");
                        continue; /* again read marks for same paper */
                }
                sum+=marks;
        }
        printf("sum of marks = %d.\n", sum);
        return (0);
}
```

5.

**a. What is Array? How to declare and initialize 1D and 2D array? Explain with example. 10M**

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.

**b. Write a C program to sort the array elements using bubble sort.** **5M**

Bubble Sort

```
/* Code Bubble Sort */
#include <stdio.h>
void bubsort(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]);
  }
  bubsort(a,tot); /* calling bubble sort */
  printf("Sorted array:\n");
  for(x=0;x<tot;x++)
        printf("%5d",a[x]);

return (0);
}
```

**c. Write a C program to implement linear search technique.** **6M**

**Ans:**

```
/*Linear search  */
#include <stdio.h>
int main()
{
    int arr[1000],i, n, skey, found=0;
    printf("Number of elements : ");
    scanf("%d",&n);
    for(i=0; i<n; i++) {
        printf("Enter element %d : ",i+1);
        scanf("%d", &arr[i]);
    }
    printf("The array:\n");
    for(i=0; i<n; i++)
        printf("%d ", arr[i]);
    printf("\nEnter element to search : ");
    scanf("%d", &skey);
    for(i=0;i<n;i++)
      if(arr[i]==skey) { found = 1; break; }
    if(found) printf("Found at %dth position\n",i+1);
    else printf("Not found\n");
  return (0);
}
```

Expected output:

```
Number of Elements: 5
Enter element 1: 56
Enter element 2: 32
Enter element 1: 67
Enter element 1: 12
Enter element 1: 44

Enter element to search: 12
Found at 4th position
```

6.

a. **What is String? Explain any 4 string library functions with syntax and example. 10M**

**Ans:**
A string is array of characters terminated with a NULL ('\0') character.
Example:

```
char name[6] = {'P','a','r','a','s','\0'};
or
char name[]="Paras";
or
char *name="Paras";
```

**String handling Library file is string.h. We include in c program as:**
#include <string.h>

Now, 4 string handling functions:
**i) strlen() : Returns length of string in bytes. This function will have one string argument.**
Example:
printf("Length of string "Dr. P. N. Singh" is %d\n", strlen("Dr. P. N. Singh"));

output : 15

**ii) strrev() : Reverses a string in its place. The function will have 1 argument.**

Example:

```
char name[] = "Sanchari";
strrev(name);
printf("%s", name);
Output: irahcnaS
```

**iii) 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
```

**iv) 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
```

**b. Write a program to multiply 2 matrices by assuming their multiplication compatibility 10M**

**Ans:**

```
/* Multiplication of matrices –   mnp concept */
#include<stdio.h>
int main()
{
   int a[100][100],b[100][100],c[100][100],i,j,k,m,n,p;
   printf("Enter total rows and cols of matrix a : ");
   scanf("%d%d",&m,&n);
   printf("Total column of matrix b : ");
   scanf("%d",&p);
   printf("Enter matrix a - %d x %d:\n",m,n);
   for(i=0;i<m;i++)
  for(j=0;j<n;j++)
      scanf("%d",&a[i][j]);
```

```
 printf("Enter matrix b - %d x %d:\n",n,p);
 for(i=0;i<n;i++)
for(j=0;j<p;j++)
   scanf("%d",&b[i][j]);

printf("Resultant Matrix:\n");
 for(i=0;i<m;i++)    {
for(j=0;j<p;j++) {
      c[i][j]=0;
   for(k=0;k<n;k++)
            c[i][j] += a[i][k] * b[k][j];
   printf("%5d", c[i][j]);
}
printf("\n");
 }
return (0);
}
```

```
Expected output:
Enter total rows and cols of matrix a : 3 4
Total column of matrix b : 4
Enter matrix a - 3 x 4:
2 3 4 5
2 1 3 4
4 3 5 6
Enter matrix b - 4 x 4 :
3 2 4 5
2 3 4 5
1 2 3 2
4 3 5 6
Resultant Matrix:
  36   36   57   63
  27   25   41   45
  47   45   73   81
```

7.

**a. What is Function? Explain different categories of user defined functions. 10M**

- A function is a self-contained block of statements that performs a coherent task of some kind. **User defined functions are building blocks of a program.**
- 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 ( ). Arguments/parameters are written within parentheses separating by , (comma).
- A function may or may not have arguments. A function may or may not return a value. If it retursn, it returns only one value.

- A void function does not return a value. Only return statement may be there to return the control.
- **A user defined function is defined by user as per task is to be performed by function. Now a days user defined functions are also called method as a function is a method to perform a task.**
- **A program is sub-divided in smaller meaningful segments. It is modular approach of programming. It reduces complexities of a larger program and the program becomes easier to check and debug.**
- 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.

**Syntax to declare a function prototype:**

**returntype functionname(type1, type2, …..);**

**Categories of User defined functions:**

**i. Not having any arguments, not returning any value**

```
#include <stdio.h>
void gao(); /* declaration*/
int main() {
    gao(); /* calling gao()*/
    return (0);
}
void gao() { /*function definition */
printf("Kudi Punjaban Dil Chura ke le gayee – Sona Sona-Dil mera Sona");
}
```

**ii) with argument does not return a value**

```
#include <stdio.h>
#define PI 3.14
void printarea(float rad) {
    printf("Area = %.3f square unit\n", PI*rad*rad);
}
int main() {
   float r,;
   printf("Enter radius of a circle : ");
   scanf("%f", &r);
   printarea(r);  /* calling printarea () function */
   return (0);
}
```

iii) without argument , returns a value

```
#include <stdio.h>
float simple_roi() {
  float r,p,t,interest;
  printf("Enter principle, years & interest : ");
  scanf("%f%f%f", &p, &t, &interest);
  r=interest/(p*t)*100;
  return (r);
}
int main() {
  float srate;
  srate=simple_roi();
  printf("Simple Rate of interest is %.2f %% yearly\n",srate);
  return (0);
}
```

iv) with argument, returns a value

```
#include <stdio.h>
#define PI 3.14
float findarea(float rad) { /* r is received as rad */
      return (PI*rad*rad);
}
int main() {
   float r, area;
   printf("Enter radius of a circle : ");
   scanf("%f", &r);
   area=findarea(r); /* calling findarea() passing r */
   printf("Area = %.3f square unit\n", area);
return (0);
}
```

v) Recursive functions : Where the function calls it self

```
/* factorial by recursion */
long int  factorial(int n) {
   if (n==0 || n==1)
          return (1);
   else
         return (n*factorial(n-1));
}
```

**b. Write a C program for evaluating the binomial coefficient using a function Factorial. 6M**

**Ans:**

Binomial Coefficient: Binomial coefficient nCr is the order of choosing ‘r’ results from the given ‘n’ possibilities. The application of this formula is applicable Pascal’s triangle:

C=fact(n)/ (fact(r)*fact(n-r));

```
/* finding binomial coefficient nCr*/
#include <stdio.h>
long int fact(int n) {
  if(n==0 || n==1)
     return (1);
  else
     return (n*fact(n-1));
}
int main() {
     int  n, r;
     long int C;
     printf("Enter value of n and r : ");
     scanf("%d%d",&n,&r);
     C=fact(n)/ (fact(r)*fact(n-r));
     printf("C = %d results of %d possibilities = %ld\n",r,n,C);
     return (0);
}
```

```
Expected output:
Enter value of n and r : 5 3
C = 3 results of 5 possibilities = 10
```

**c. Explain Local and Global variables with example** **4M**

**Ans:**

**Global – storage class extern:**

- Identifiers defined outside of a function are by default gobal (whether extern is written or not).
- They will be publically available to all the functions and may be corrupted/modified by any of the functions.

**Local – storage class auto:**

- Identifiers defined inside  a function are by default local (whether auto is written or not).
- **No way local variables defined within a function will be modified by other function.**
- **Scope of a local variable is within the block only or the function in which it is defined**

```
#include <stdio.h>
extern int g = 100;
void func(int a, int b) {
   printf("in func Value of global g = %d\n",g);
   printf("In func a = %d b = %d\n",a,b);
   a=40; b= 50; g=200;
   printf("In func changed value a = %d b=%d g = %d\n",a,b,g);
}
int main() {
  int a=20,b=30; /* auto int a= 20, b=30 */
  printf("In main a = %d b=%d g = %d\n",a,b,g);
  func(a,b); /* calling func with a=20 & b= 30 */
  printf("in main after function call a = %d b= %d g=%d\n",a,b,g);
  return (0);
}
```

**Other 2 storage class are local: static and register**

8.

**a. Differentiate i) User defined and built in function ii) Recursion and Iteration.** **10M**

**Ans:**

**i) User defined and built-in functions**

- Functions are classified of two types – In-built function and User/Programmer defined functions. A function name is followed by a pair of ( ).  Arguments are written within ().

Built-in functions are already defined, compiled and stored in library files. The library file math.h have several defined built-in functions for math operations like sqrt(), pow(), sin(), cos(). These functions cannot be modified by the user. For example, sqrt() to find the square root of a number:

```
#include <stdio.h>
#include <math.h>
int main ()  { printf("Square root of 2 = %f\n", sqrt(2)); return 0; }
/* will give the output square root of 2 i.e. 1.414…. */
```

Functions defined by programmers in a program is known as user-defined or programmer-defined functions. These function can be modified by the programmers. User understands working of these functions.

Example:

```
void gao() { printf("Kudi Punjaban Dile chura ke lae gayee – Sona Sona"; }
```

Here gao() is user defined functions.

**ii) Recursion and iteration:**

Recursion is a process of calling a function by itself repeatedly.

5! = 5*4*3*2*1

It can be written as:

5! = 5*4!

Recursive call of a function is permitted in programming language.

A function may or may not have arguments. Arguments are written within parentheses separating by , (comma).

```
/* factorial by recursion */
long int  factorial(int n) {
   if (n==0 || n==1)
           return (1);
   else
          return (n*factorial(n-1)); /* here factorial function calls it self---Recursion*/
}
```

**Iteration is used in looping statements where a block of statements repeatedly execute till condition is satisfied (evaluated true).**

There are three iterative statements in C: for, while and do…while

Syntax of an iterative statement while:

```
while (condition evaluates true)
{
   /* block of statements ; */
}
```

Example to print odd numbers from 1 to 100:

```
int n=1;
while(n<=100) {
    printf("%d\n", n);
    n=n+1;
}
```

**b. Explain Call by value and Call'by,re ference with example. 10M**

**Ans:**

- **Call by value:** A function is called by passing value or variable containing the value. Copy of data is made and stored by the way of name of the parameter/argument. Any changes to the parameter have NO effect in the calling function

```
/* call by value */
#include <stdio.h>
void func1(int a, int b) {
    int temp=a;
    a=b;
    b=temp;
    printf("In function a=%d b=%d\n", a, b);
}
int main() {
  int a=20, b=30;
  printf("In main a= %d b= %d\n",a,b);
  func1(a, b); /* call by value */
  printf("After function call in main: a = %d b = %d\n", a,b);
  return (0);
}
```

**Expected output:**

```
In main a= 20 b= 30
In function a= 30 b = 20
After function call in main: a = 20 b =30
```

- **Call by reference: A function is called by passing the reference (address). This parameter refers to the original data in the calling function. Thus changes made to the parameter are also made to the original variable of calling function.**
- **& is used for address/reference**
- **Arrays, structures are by default passed as reference.**

```
/* call by reference */
#include <stdio.h>
void swap(int *p, int *q) {
    int temp=*p;
    *p = *q;
    *q=temp;
    printf("In function values are swapped: %d and %d\n", *p, *q);
}
int main() {
  int a=20, b=30;
  printf("In main a= %d b= %d\n",a,b);
  swap(&a, &b); /* call by reference */
  printf("After function call in main: a = %d b = %d\n", a,b);
  return (0);
}
```

**Expected output:**

```
In main a= 20 b= 30
In function values are swapped: 30 and 20
After function call in main: a = 30 b =20
```

9.

a. **What is Structure? Explain Structure declaration and Initialization with example. 10M**

**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.

**Syntax:**

```
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];
```

**Example:** Structure declaration:

```
   struct student
   {
           int roll;
           char name[20];
           int marks;
};
```

**Structure variable declaration:**

**struct student s1,s2;**

**Structure initialization examples:**

With declaration:

```
   struct student
   {
           int roll;
           char name[20];
           int marks;
} s1= {111, "Pratap", 76};
```

**By individual members:**

**s2.roll = 222;**

**strcpy(s2.name, "Rubiya");**

**s2.marks=89;**

**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**

**b. What is union? How to declare union? List out the differences and similarities between structure and union. 10M**

**Ans:**

**Union is special derived data type of that allows to store different data types in same location. It may provide the same memory location for multiple purpuses.**

```
union tagname
{
   data_type member_1;
   data_type member_2;
   data_type member_3;
   ...
   data_type member_N;
};
```

Example to declare and initialize:

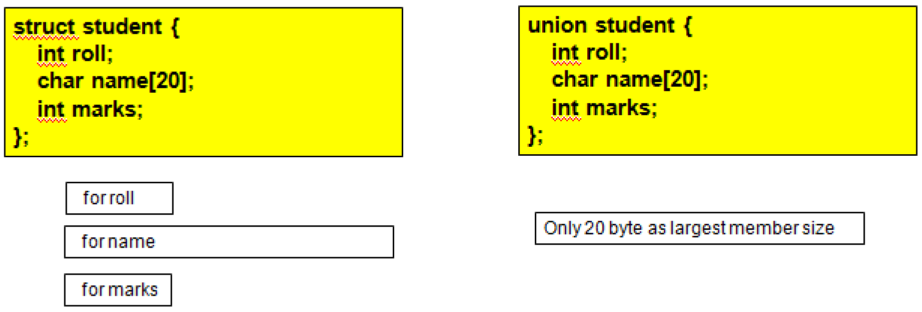
union student { int roll; char name[30]; int marks; };

union student s1 = {56};

Here firsr member of the union i.e. s1.roll is initialized. We can use it and then we can access others one by one.

**Difference in struct and union:**

The important difference between structures and unions is that in structures each member has it's separate memory locations but for union is declared the compiler allocates only one memory sufficient to hold the largest member of the union. Members of the union can be accessed one by one only and only one value will be there.



```
struct student {
   int roll;
   char name[20];
   int marks;
};
```

```
union student {
   int roll;
   char name[20];
   int marks;
};
```

10.

**a. What is Pointer? How to declare and initialize pointers? Explain with example. 6M**

**Ans:**

A pointer keeps address of data. A pointer is variable that represents the location ( rather than the value) of a data item, such as a variable or an array element.

Pointers is an important feature of C and frequently used in C. They have a number of useful applications.

Pointer operator is asterisk ( * ).

To declare a pointer: *datatype *pointervariable;*

To initialize: pointervariable = &data;

& is address of operator.

Example1:

```
int n;
int *p;  /* here p is pointer variable & can be used to keep address */
p=&n;  /* here p keeps address of n */
```

Example2:

```
int a[5] = { 10,20.30,40,50};
int *p;
p = a;   /* It is equivalent to  p = &a[0];   Here it keeps address of first element */
```

```
/* Pointers in C */
#include <stdio.h>
main()
{
 int n,*p;   /* n is an integer variable, *p is a pointer variable */
 n=20;      /* value 20 is assigned to variable n */
 p=&n;      /* address of n is assigned to p. & is address of operator */
 printf("Value of n = %d\n",n);
 printf("Address of n = %u\n",p);
 printf("Value stored at address = %d\n",*p);
          /* * is de-referencing/indirection operator*/
 *p=40;     /* value changed at address */
 printf("changed Value of n = %d\n",n);
 printf("Value of *p = %d\n",*p);
 n=80;      /* value changed by variable */
 printf("Value stored at address = %d\n",*p);
 return (0);
}
```

```
Output
Value of n = 20
Address of n = …… (memory address)
Value stored at address = 20
changed Value of n = 40
Value of *p = 40
Value stored at address = 80
```

**b. Write a C program to find sum of two squared number using Macro square (n). 6M**

Ans:

```c
/*Sum 2 squared number using macro square n */
#include <stdio.h>
#define square(n) (n)*(n)
int main()
{
  int a,b;
  printf("Enter 2 numbers : ");
  scanf("%d%d", &a,&b);
  printf("Sum of squared numbers = %d\n", square(a)+square(b));
  return (0);
}
```

Expected output:

```
Enter 2 numbers : 5 9
Sum of squared numbers = 106
```

**c. Write a C program to find sum, mean, standard deviation of array elements using pointers. 8M**

Ans:

```c
/* Sum, mean and Standard deviation */
#include <stdio.h>
#include <math.h>
#define size 5
int main() {
 int x[size],*p,sum=0,i,diff,summd;
 float mean,variance,stdev;
 p=x;
 for(i=0;i<5;i++) {
   printf("Enter Number %d : ",i+1);
   scanf("%d", p+i); /* we can write &*(p+i) */
   sum+= *(p+i);
 }
 mean= (float)sum/size;
 summd=0;
 for(i=0;i<5;i++) {
   diff=mean- *(p+i); /*mean - x[i] */
   summd+= (diff*diff);
 }
 variance=(float)summd/(size-1);
 stdev=sqrt(variance);
 printf("Sum = %d Mean = %.2f Standard Deviation = %.2f\n",
        sum,mean,stdev);
 return (0);
}
```

Expected output:

```
Enter Number 1 : 5
Enter Number 2 : 6
Enter Number 3 : 7
Enter Number 4 : 8
Enter Number 5 : 9
Sum = 35 Mean = 7.00 Standard Deviation = 1.58
```