Important Instructions to examiners:
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q.1.   **Attempt any TEN of the following:**   **20**
     a) **List the types of constants.**
       (½ mark for each type)
       1) Integer constants
       2) Real constants
       3) String and character constants
       4) Backslash constants

     b) **List bitwise operators and explain any two.**

       (List containing any 4 bitwise operators – 1 mark, explanation of any two – 1 mark)
       Bitwise operators:
       | – Bitwise OR
       & – Bitwise AND
       ~ – Bitwise NOT
       ^ – Bitwise XOR
       << – left shift
       >> – right shift
       Explanation:
       Bitwise OR – |
       It takes 2 bit patterns, and performs OR operations on each pair of corresponding bits. The following example will explain it.

```
        1010
        1100
     --------
OR      1110
```

       Bitwise AND – &
       It takes 2 bit patterns, and perform AND operations with it.

```
        1010
        1100
     ------
AND     1000
     -------
```

The Bitwise AND will take pair of bits from each position, and if only both the bit is 1, the result on that position will be 1. Bitwise AND is used to Turn-Off bits.

Bitwise NOT

One's complement operator (Bitwise NOT) is used to convert each "1-bit to 0-bit" and "0-bit to 1-bit", in the given binary pattern. It is a unary operator i.e. it takes only one operand.

```
1001
----    NOT
0110
-------
```

Bitwise XOR ^

Bitwise XOR ^, takes 2 bit patterns and perform XOR operation with it.

```
        0101
        0110
       ------
XOR     0011
       ------
```

Left shift Operator – <<

The left shift operator will shift the bits towards left for the given number of times.
```
int a=2<<1;
```

Right shift Operator – >>

The right shift operator will shift the bits towards right for the given number of times.
```
int a=8>>1;
```

c) **State the use of continues statement. (explanation 2 marks)**

- The continue statement is used to transfer the control at the beginning of loop
- Whenever the continue statement is encountered immediately the control passes at the beginning of the loop continue is generally used with if statement.

Example:
```
#include <stdio.h>
#include <conio.h>
void main()
{
int i,j; clrscr ( );
for (i=1; i<=2; i++)
{
printf("\n");
for (j=1; j<=2; j++)
{
if (i= =j) continue;
printf ("\n %d %d\n",i,j);
}
getch ( );
}
```
Output:
1 2
2 1

d) **Write output of following program:**
```
#include<stdio.h>
void main()
{
Intchoice=0 switch (choice)
{
Case0; printf("Mumbai");
Case1; printf("Pune");
}
getch();
}
```
(Correct output – 2Marks)

OUTPUT :

MumbaiPune

e) **State the use and syntax of strcmp() function.**
(use – 1 mark, Syntax – 1 Mark)
strcmp():- This function compares two strings identified by arguments and returns zero if both strings are equal, otherwise it returns the difference between ASCII values of first non matching character pair from the strings.
Syntax: strcmp (string1, string2);
strcmp ("there", "their") returns value as 9 which is the difference between r' & 'e'

f) **Find error in following program and justify it:**
```
#include<stdio.h>
void main ()
{
Int i, a[5]={7, 5, 2,1,9,14};
For (i=0; i<5; i++)
Printf("%f", a[2]);
getch();
}
```
(1 mark for each error with justification. Output not necessary)
Errors :
1) array size is 5 and number of variables initialized is 6.
2) printf statement should include %d as control specifier because data type of array a is int.

Output :
22222

g) **State call by value with example.**

(Explanation – 1 mark, example – 1 Mark)

When a function is called by passing actual parameters in function name then it is called as call by value. In call by value method actual parameters are copied to formal parameters. The called function works on the copy and not on original values of the parameters. Because of this the original data cannot be changed accidently.

Example:

```
#include<stdio.h>
#include<conio.h>
main()
{
int p,a,b;
printf("Enter two numbers:";
scanf("%d %d",&a,&b);
p=product(a,b);
}
product(int x,int y)
{
int p1=1;
p1=x*y;
return(a,b);
}
```

h) **State the need of function.**

(Explanation -2 marks)
A function is a self contained block of statements which is used to perform a specific task.
Need:
1. It is possible to code any program utilizing only main function but it leads to a number of problems.
2. The program may become too large and complex and as a result the task of debugging, testing and maintaining becomes difficult.
3. If a program is divided into functional parts, then each part may be independently coded and later combined into a single unit.
4. These independently coded programs are called as subprograms that are much easier to understand, debug and test.

i) **What is the use of indirection operator (*)?**

(use – 1 mark, example – 1 mark)

Indirection operator * is used in 'C' language to get the value stored at the address pointed by any pointer variable. It can also be used in pointer declaration just to indicate the type of the variable is a pointer type.
 For example :

1) int a, *ptr;   //here a is any integer variable and ptr is an integer pointer.
2) a=5;   // a is assigned value 5.
3) ptr=&a;   // address of a is stored in ptr
4) printf("%d   %d", a, *ptr);   // will print the output as 5 5 as *ptr uses indirection operator to get the value stored at address pointed by ptr.

j) **Write the syntax of nested if else statement.**
   (Correct syntax – 2 Marks)
   Nested if else statement:
   When a series of decisions are involved, we may have to use more than one if…else statement in nested form as shown below:

```
Syntax:
if(test condition1)
  {
    if(test condition2)
      {
         statement-1;
      }
    else
      {
         statement-2;
      }
  }
else
  {
         statement-3;
  }
statement-x;
```

k) **State four rules for choosing variable name.**
   (Any 4 valid rules – 1/2 Mark each)
   Four rules for choosing variable name:
   i. It must begin with a letter. Some systems permit underscore as the first character.
   ii. ANSI standard recognizes a length of 31 characters. However length of variable should not normally more than eight characters, since only the first characters are treated as significant by many compilers.
   iii. Uppercase and lowercase are significant. That is, the variable Total is not the same as total or TOTAL.
   iv. Variable should not be a keyword.
   v. In variable declaration white space is not allowed.

l) State two difference between switch statement and if statement.
(Any valid 2 differences – 1 Mark each)

Switch() statement	If statement
1. Switch case can be used when there are multiple options and only one is to be selected.	1. If statement checks any number of conditions and takes the action for true case and false case.
2. Switch statement operates only on equality of values.	2.If works even with the other comparison operators like >,<,>=,<=
3. Switch has a default case.	3. If does not have any provision to handle default case.
4. Every case needs a break statement to come out of switch block once a condition is matched.	4. No such requirement in case of 'if' statement.

Q.2. Attempt any FOUR of the following: 16
a) Explain increment and decrement operator with example.
(Explanation of each operator – 1 mark each, example of each operator – 1 mark each)
Increment operator:
   i. Increment operator (++) is an unary operator. It operator a one operand.
   ii. It is used to add one to list operand.
      Example:
```
# include <stdio.h>
# include<conio.h>
void main()
{
int a=6;
clrscr();
printf("%d",a);
a++;
printf("\n%d",a);
getch ();
}
```
      In above example due to increment operator (++) value of a will became 7.

Decrement operator:
   i. Decrement operator(--) is an unary operator. It operates on one operand.
   ii. It is used to subtract one from its operand.
      Example
```
# include <stdio.h>
# include<conio.h>
void main()
{
int a=5;
clrscr();
printf("%d",a);
a--;
printf("\n%d",a);
getch ();
}
```
      In above example due to decrement operator (--) value of a will become 4.

b) **Explain formatted input and formatted output with example.**
(Explanation of scanf() and printf() – 1 Mark each, example of each – 1 Mark)
Formatted input: - formatted input refers to an input data that has been arranged in a particular format.
To read formatted input c provides scanf() function.
Syntax :

scanf("control_string 1, - - , control_string n",&variable1, - - - - ,variable n);

where control_string is as per the data type of variable used in the later part of the syntax.
&variable shows the address at which value of variable is getting stored.
Every data type has its unique control specifier as %d for int, %f for float, %c for char etc.
eg: To input values to three variables rollno,name of studentr and percentage, scanf() can be written as
scanf("%d %s %f", &rollno, name, &percentage);

Formatted output: - formatted output refers to printing the message & numerical values of the variable.
To print formatted output „C" provides printf() function.
Syntax :

printf("control_string 1, - - , control_string n",variable1, - - - - , variable n);

where control strings can be same as explained earlier.
eg : To print values to three variables rollno,name of studentr and percentage, printf() can be written as
printf("%d %s %f",rollno, name, percentage);

c) **Enlist the types of token available in C and give one example for each.**

(Any 4 valid types with one example – 1 Mark each)
C tokens are of six types. They are,
1) Keywords eg: while,
2) Identifiers eg: total
3) Constants eg: 10
4) Strings eg: "hello"
5) Special symbols eg: (), { }
6) Operators eg: +, /,-,*

d) **State the importance of flowcharts with the symbols used for drawing flowchart.**
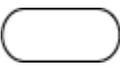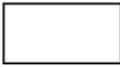**(Importance – 2 Marks, Symbols – 2 Marks)**
Sometimes it's more effective to visualize something graphically that it is to describe it with words.
Flowchart is the tool which helps to represent logic of the program pictorially.
Flowcharts explain a process clearly through symbols and text.
Symbols used in flowchart :

1)   Oval - Start/ Stop

2)   Rectangle - Process

3)   Parallelogram - Input/ Output

4) ⬦   Diamond - Decision
5) -------->   Arrow - Flow lines

e) Write a program to print following pattern:
   1
   2 2
   3 3 3
   4 4 4 4
   5 5 5 5 5
   (Correct Logic – 2 Mark, Correct syntax – Marks)

```
# include <stdio.h>
 # include <conio.h>
 main ()
{
    int i, j ;
    for ( i=1; i<=5; i++)
    {
        for(j=1; j<=i; j++)
        {
         printf("%d", i);
        }
         printf("\n");
    }
}
```

f) State any four differences between while loop and do while loop.
   (Any other valid difference can be considered. Each difference 1 M (4 Diff.))

While loop	do while loop

1. Syntax: while (expression)
     {
      Statements
     }
	syntax: do
	     {
	      Statements
	     }
	     while (expression);

2. while loop checks the condition first and then executes statements.	do while loop first executes statements and then checks the condition.
3. If expression is true then only statements inside block are executed otherwise loop terminates.	If expression is false then also at least once statements inside block are executed.

4. Example: a=1;
 while (a<0)
 {
   printf ("%d",a);
   a = a+1;
 }
	a = 1;
	 do
	 {
	   printf("%d",a);
	   a = a+1;
	 }while(a<0);

Q.3.   Attempt any FOUR of the following: 16

a) **Write a program to find the sum of digits of an integer (e.g input=1453 sum=1+4+5+3=13)**
(Correct Logic – 2 Mark, Correct syntax – Marks)

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int num=0,rem=0,sum=0;
clrscr();
printf("enter number");
scanf("%d",&num);
while(num!=0)
{
rem=num%10;
sum=sum+rem;
num=num/10;
}
printf("Sum of Digit is %d",sum);
getch();
}
```

b) **Write a program to find largest of three numbers using nested if-else.**
(Correct Logic – 2 Mark, Correct syntax – 2 Marks)

```c
#include <stdio.h>
#include<conio.h>
int main()
{
   int a,b,c;
   clrscr();
   printf("Enter 3 numbers: \n");
   scanf("%d %d %d",&a,&b,&c);
   if(a>=b)
   {
      if(a>=c)
         printf("%d is largest",a);
      else
         printf("%d is largest",c);
   }
   else if(b>=c)
      printf("%d is largest",b);
   else
         printf("%d is largest",c);

 getch();
}
```

c) **State use of goto statement with example.**
(Explanation 2M; Example 2M)
Go to statement :
i. Go to statement is used to control the flow of program as per users wish.
ii. It is used to provide branching of conditions.
iii. Goto transfers the control of the program from one statement to another
iv. The statement where control transferred is labeled by a label i.e. the name is given to that particular section or a single statement.
Example:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
 int no;
 clrscr ();
accept: printf ("Enter any Number: ");
scanf ("%d",&no);
 if (no= =0) go to accept;
getch ( );
 }
```

In the above example if user inputs a number 0 then using go to it goes to accept.

d) **Write a program to find factorial of a number using for loop.(n=1)**
(Correct Logic – 2 Mark, Correct syntax – Marks)

```
#include<stdio.h>
 #include<conio.h>
void main()
{
intn,i;
int fact=1;
clrscr();
printf("enter value of n:");
scanf("%d",&n);
for(i=1;i<=n;i++)
 {
fact=fact*i;
}
printf("factorial of number is %d",fact);
getch();
}
```

e) **Explain switch statement with flow chart.**
(Explanation 2M; Flow chart 2M)
Switch statement accepts value & tests it for various conditions. If all conditions are false then the default statement will get execute.
The diagram shows pictorial representation of execution of switch case loop. The conditions are checked by one by one.
The statement of the case would be execute if the condition is true. Else control pass to the next case.
 The different conditions in switch are called as cases. These cases are break using break keyword. If any one of the case is true then statements written in the same case would get execute & the case will get due to break keyword.

example: Accept character form user & check character is vowel or not using switch statement.

```c
#include<stdio.h>
#include<conio.h>
void main()
{
char code;
clrscr();

printf("\n Enter character:");

scanf("%c",&code);

switch(code)                    //passing code to check the value

{

case 'a':

 case 'e':


case 'i':                       // all cases a,e,i,o u are single character so enclosed in

                                single quotes

 case 'o':

 case 'u':

 printf("%c is vowel",code);
break;
default:
printf("the %c is not vowel",code);
break;
}
getch();
}
```
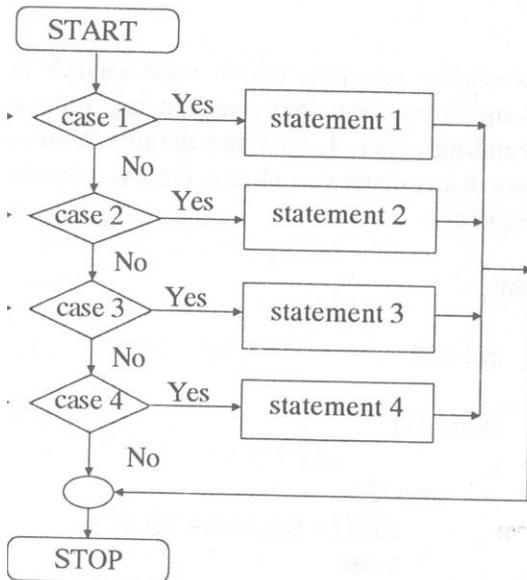
O/P:
Enter character: s
s is not vowel

f) **Write a program to find whether the entered number is prime or not.**
(Correct Logic – 2 Mark, Correct syntax – Marks)

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int num;
clrscr();
printf("enter number");
scanf("%d",&num);
if(num==1||num==2||num==3||num==5||num==7)
{
printf("\n\n%d is prime",num);
}
else if(num%2>0&&num%3>0&&num%5>0&&num%7>0)
{
printf("\n\n %d is prime",num);
}
else
{
printf("\n %d is not prime",num);
}
getch();
}
```

Q.4. **Attempt any FOUR of the following:** **16**

a) **Declare and initialize the one dimensional array with 10 elements. Explain how the elements in an array can be accessed.**

Declaration of Array:
Syntax: datatype variable-name[size]

Declaration of 10 array element is : **(1M)**
Int a[10];
Where a is variable name or array name, 10 is size of an array, int is datatype

Initialize the one dimensional array: **(1M)**
Syntax: datatype array-name[size] ={list of values};

Initialization of 10 array elements:
Int a[10]={ 10,20,30,40,50,60,71,70,80,90};

Accessing Element of an Array: (2M)
Once an array is declared, its individual element in the array can be referred. This is done with the subscript number in the brackets following the array name . this number specifies the element's position in the array. All the array element are number starting with '0'. Thus marks[2] is not the second element of array but the third.

b) **Differentiate between character array and integer array with respect to size and initialization.**
(Any Four each 1Mark)

Sr.No	integer array	character array
1	An integer array is an array whose elements are all of an integer type which has no fractional component.	character array is an array which contains nothing but character types
2	An integer array is capable of holding values of type int.	A char array is capable of holding values of type char
3	int array take more memory to allocate	char array take less memory to allocate
4	integer array takes 2 bytes for each cell	whereas char takes 1 byte for each cell
5	array 's r not terminated with NULL(\0) character	Strings r terminated with NULL(\0) character

c) **Write a program to sort elements of an array in ascending order.**
   (Correct Logic – 2 Mark, Correct syntax – Marks)

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

{
int i, j, a, n, number[30];
clrscr();
printf("Enter the size of an array \n");
scanf("%d", &n);
printf("Enter the array elements \n");
for (i = 0;i<n; i++)
scanf("%d", &number[i]);
for (i = 0; i < n; i++)
{

for (j = i + 1; j < n; j++)
        {

            if (number[i] > number[j])

        {
                a =number[i];
                number[i] = number[j];
                number[j] = a;

        }

      }

    }

    printf("The numbers arranged in ascending order are \n");

    for (i = 0; i < n; i++)

        printf("%d\n", number[i]);
getch();
}
```

d) **State four storage classes. Explain any one.**

There are four storage classes in C
1) **Auto**
2) **Static**
3) **Extern**
4) **Register**
   (State four storage classes 2M; Explanation of any storage class 2M)
1) **Auto:** Automatic variables are generally declared in the function in which they get utilized. They are known as local variable. Automatic variable can be created when function is called and get destroyed when execution get function over.
   We can declare variable with auto as follows
   main()
   {
   auto int number;
   ……….
   ……….
   }

2) **Static:** The declaration of the variable is done using the static keyword. The value of static variable remains constant throughout the program. Variable declaration can be done as follows

   static int i;

3) **Extern:** External variable are global variable which are declare above the main() function. Since they are global, they can accessible to all the function in the program. Scope of these variable throughout the program.

   {
   extern int a;//declarations
   …..
   …..
   }
   int a   //definition

4) **Register:**
   The declaration of the variable is done using the register keyword. Then instead of storing variable in the normal memory it stores variable in the machine's register.
   Main()
   {
   Register int i=1;   //declaration of register variable
   ……
   ……
   }

e) **Write a program to perform addition, subtraction, multiplication and division of two integer numbers using function.**
   (Correct Logic – 2 Mark, Correct syntax – Marks)

```c
#include<stdio.h>
#include<conio.h>
void add(int p,int q);
void subtraction (int p,int q);
void multiplication(int p,int q);
void division(int p,int q);
void main()
{
int m,n,sum,sub,mul,div;
clrscr();
printf("enter two integer numbers::");
scanf("%d %d",&m,&n);
add(m,n);
subtraction(m,n);
multiplication(m,n);
division(m,n);

getch();
}
void add(int p,int q)
{
int sum;
sum=p+q;
printf("Addition of a numberis %d\n",sum);

}
void subtraction(int p,int q)
{
float sub;
sub=p-q;
printf("\n subtraction of a number %f",sub);
}
void multiplication(int p, int q)
{
int mul;
mul=p*q;
printf("\nmutiplication of a number=%d\n",mul);
}
void  division(int p,int q)
{
float div;
div=p/q;
printf("\n Division of a number=%f\n",div);
}
```

f) Write a program to declare a structure student having data member, roll_no; name ans agg_marks. Accept data and display this information for one student.
(Correct Logic – 2 Mark, Correct syntax – Marks)

```
#include<stdio.h>
#include<conio.h>
struct student
{
char name[20];
floatagg_marks;
Introll_no;
};
void main()
{
struct student s;
clrscr();
printf("Enter the name of a student:");
scanf("%s",s.name);
printf("\nEnter the aggregate marks:");
scanf("%f",&s.agg_marks);
printf("\nEnter the roll number:");
scanf("%d",&s.roll_no);
printf("\n\nStudent data is:");
printf("\nName=%s",s.name);
printf("\nAggregate marks=%f",s.agg_marks);
printf("\n Roll number=%d",s.roll_no);
getch();
}
```

Q.5.   Attempt any FOUR of the following: 16

a) State two advantages and two disadvantages of array.

(Advantages 2 marks , disadvantages 2 marks any two valid points each)
Advantages:-
1. Not necessity to declare two many variables.
2. Arrays permit efficient (constant time, O(1)) random access
3. not efficient insertion and deletion of elements (which are O(n), where n is the size of the array)
4. Arrays also are among the most compact data structures; storing 100 integers in an array takes only 100 times the space required to store an integer, p
Disadvantages :-
1. Array is static structure. It means that array is of fixed size. The memory which is allocated to array can not be increased or reduced.
2. Since array is of fixed size, if we allocate more memory than requirement then the memory space will be wasted. And if we allocate less memory than requirement, then it will create problem.
3. The elements of array are stored in consecutive memory locations. So insertions and deletions are very difficult and time consuming .

b) **Write a program for addition of two 3 X 3 matrices.**

(Logic – 2 Marks, Syntax-2 Marks)

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int a [3][3], b[3][3],c[3][3],i,j;
printf("Enter First matrix ");
for (i=0;i< 3;i++)
{
for (j=0;j<3;j++)
{
scanf("%d",&a[i][j] );
}
}
printf ("Enter second matrix:" );
{
scanf("%d",&a[i][j] );
}
}
printf ("Enter second matrix:" );
for (i=0;i<3;i++)
{
for (j=0;j<3;j++)
{
scanf ("%d", &b[i][j]);
}
}
for (i=0;i<3;i++)
{
for (j=0;j<3;j++)
{
c[i][j]=a[i][j]+b[i][j];
   }
   }
printf ("The resultant matrix is;");
for (i=0;i<3;i++)
{
for (j=0;j<3;j++)
{
printf ("\t%d",c[i][j] );
   }
printf ("\n" );
}
getch();
        }
```

c) **Define recursive function. List any two advantages of recursive function.**
(Definition 2 marks advantages two marks)

Recursive Function:-
Recursion is a process of function calling itself again and again. A recursive function contains function call to itself in the body of function.
Advantages of recursive function.
1. Reduce size of the code when you use recursive call.
2. Recursion is used to divide the problem into same problem of subtypes and hence replaces complex nesting code.
3. Using recursion we can avoid unnecessary calling of functions.

d) **Discuss function prototype with syntax. List the four categories of function.**

(protyping 2 marks, categories 2 marks)
Function prototype is a function declaration before it is invoked. It consists of function return type, function name, parameter list and terminating semicolon.
Syntax: Function_typefunction_name(parameter list);
function_type specifies the data type of value that the function is expected to return to the calling function. Function name specifies a unique identity of the defined function in the program. Parameter list declares the variables that will receive data sent by calling function. They act as input to the function.
Different categories of function with its prototype
1) Function with no arguments and no return values.
2) Function with arguments and no return values.
3) Function with no arguments and return values.
4) Function with arguments and return values.

e) **Write a function to print Fibonacci series up to 10 terms.**

(Logic – 2 Marks, Syntax-2 Marks)
```
#include<stdio.h>
#include<conio.h>
void main()
{
int c,p,count=10,i,n;
c=0;
p=1;
printf("0");
for(i=0;i<=count-2;i++)
        {
        n=p+c;
        p=c;
        c=n;
        printf("\t%d",c);
        }
getch();
}
```

f) **Define structure and array of structure. Declare a structure student with element roll no and name.**

(Structure definition & array of structure definition 1 mark each declaration of structure student 2 marks)
Structure:-
A structure is a collection of one or more variables of same or different data types grouped together under a single name.
Array of structure:-
arrays of Structures; arrays of structures mean collection of structures, in other word array storing different type of structure member variables.

```
Struct student
{
 int rollno;
char name[20];
};
```

Q.6. **Attempt any FOUR of the following:** **16**

a) **Write a program to declare a structure book having data members, title, author and price. Accept data and display information for one book.**

(Logic – 2 Marks, Syntax-2 Marks)
```
#include<stdio.h>
#include<conio.h>
struct  book
        {
        char tit[20];
        charauth[20];
        int price;
        }b1;

void main()
{
printf("enter title of book");
scanf("%s",b1.tit);
printf("enter author name of book");
scanf("%s",b1.auth);
printf("enter price of book");
scanf("%d",&b1.price);
printf("detdails of book are");
printf("\n\t title:%s\n\t author:%s\n\t price:%d",b1.tit,b1.auth,b1.price);
getch();
}
```

Summer – 15 EXAMINATION

b) Define pointer. How the pointers are declared and initialized?

(Definition 2 marks declaration and initialization 1mark each)

Pointers are variables used to store the memory addresses of the variables. Variables store the values and pointers stores their addresses at which these variables are located. The values stored in the pointers are integer values.

Pointer declaration & intilisation
Likewise other variables pointer also needs to be declared.
In declaration statement of pointer name of pointer variable is preceded by * (indirection operator) operator .
<u>Syntax:-</u>

data_type * name_of_variable

Ex.
int *ptr ;
Here ptr variable of data type integer pointer is declared as name of variable 'ptr' is preceded by * (indirection operator) means that variable ptr is a pointer variable..
Data type of pointer variable should similar to data type of variable of which it is going to store address
Pointers are initialized by address of another variable& for accessing address of a variable "**address of** " operator is used . this operator will fetch address of variable to which it prefix (precedes) .

Ptr=&name_of_variable;

c) Write a program to exchange values of two variables using pointer.
(Logic – 2 Marks, Syntax-2 Marks)

```
#include<stdio.h>
#include<conio.h>
void main()
       {
       int a,b,*p1,*p2,temp;
       clrscr();
       printf("enter value of a");
       scanf("%d",&a);
       printf("enter value of b");
       scanf("%d",&b);
       printf("\nvalues before exchange are a=%d b=%d",a,b);
       p1=&a;
       p2=&b;
       temp=*p1;
       *p1=*p2;
       *p2=temp;
       printf("\nvalues after exchange are a=%d b=%d",a,b);
       getch();
       }
```

d) State four features and advantages of pointers.
(2 marks for features 2 marks for advantages (two points each))
Features:-
- Pointers provide direct access to memory
- Pointers provide a way to return more than one value to the functions
- Pointers can be used to pass information back and forth between the calling function and called function.
- Pointers allow us to perform dynamic memory allocation and deal location.
- Pointers helps us to build complex data structures like linked list, stack, queues, trees, graphs etc.

Advantages:-
- Reduces the storage space and complexity of the program
- Reduces the execution time of the program
- Provides an alternate way to access array elements
- Pointers allow us to resize the dynamically allocated memory block.
- Addresses of objects can be extracted using pointers
- Increase access speed

e) Write program to find product of two no's using pointer.
(Logic – 2 Marks, Syntax-2 Marks)
```
#include<stdio.h>
#include<conio.h>
void main()
{
int *p1,*p2,a,b,mul;
clrscr();
printf("enter first no.");
scanf("%d",&a);
printf("enter second no.");
scanf("%d",&b);
p1=&a;
p2=&b;
mul=(*p1)*(*p2);
printf("\nmultiplication is %d",mul);
getch();
}
```

f) Explain meaning of following statement with reference to pointers:
Int*p,x;
X=10;
***p=x;**
P=&x;
(2 marks for *p=x & 2mks for p=&x explanation)
Int *p,x;
X=10;
*p=x;
P=&x;

Given statements   pointer p is declared. Value 10 is assigned to variable x .then content of x are copied to address which is currently held in p (garbage address) .in last statement address of x is copied in pointer p