KLS

Vishwanathrao Deshpande Institute of Technology
Haliyal - 581329

Subject : Introduction to C-programming
Model Question Paper Scheme & Solution
2025-26

Subject code : 1BPLC105E/205E

Department : Computer Science & Engineering

Sem : I/II

Staff Name : Prof. Shree Gowri SS - [illegible]

HOD : Dr. Venkatesh S - [illegible]

HOD
Computer Science & Engineering
KLS Vishwanathrao Deshpande
Institute of Technology, Haliyal.

Dean Academic : Dr. Gururaj. H - [illegible]

Dean, Academics
KLS VDIT, HALIYAL

Model Question Paper- I

CBCS SCHEME

First/ Second Semester B.E Degree Examination, 2025-26

Introduction to C Programming (1BPLC105E/205E)

TIME: 03 Hours **Max.Marks:100**

Notes:

1. *Answer any FIVE full questions, choosing at least ONE question from each MODULE*
2. *M: Marks, L: Bloom's level, C: Course outcomes.*

Module -1			**M**	**L**	**C**
Q.01	a	Define algorithm. Outline an algorithm to convert temperature from Fahrenheit to Celsius.	8	L2	CO1
	b	Define functions. Explain basic structure of a C Program.	5	L2	CO1
	c	Explain Input and Output functions in C Programming with suitable example	7	L2	CO1
OR					
Q.02	a	Explain the role of flow chart in program development. List the symbols used in designing a flowchart. Illustrate with one example.	8	L2	CO1
	b	List the features of C programming language. Explain the process of compiling and executing a C Program.	5	L2	CO1
	c	Define Identifier. List the rules for framing an identifier with an example to each rule.	7	L2	CO1
Module-2					
Q. 03	a	Explain the following operators i) Increment and Decrement operators. ii) Logical Operators	8	L2	CO1
	b	Explain the else if ladder with syntax and a suitable program.	6	L2	CO2
	c	Outline the syntax of switch statement. Given an integer between 1 to 7 representing the day of week, develop a program to display day in words using switch statement [E.g. for a number 1, print Sunday and for number 7, print Saturday]	6	L3	CO2
OR					
Q.04	a	Show the evaluation order of the following expression with intermediate and final values: 100/20<=10-5+100%10-20==5>=1!=20	8	L2	CO1
	b	Differentiate between entry controlled loop and exit controlled loop.	6	L2	CO2
	c	Develop a C program to find the sum of first n numbers using while loop.	6	L3	CO2
Module-3					
Q. 05	a	Define string. Develop a program to read a string, reverse the string and print.	6	L3	CO3
	b	Define array. List and explain the types of arrays.	6	L2	CO3
	c	Develop a program to multiply two N*N matrices.	8	L3	CO3
OR					
Q. 06	a	Develop a program to find the length of a string without using built in function.	6	L3	CO3
	b	Explain declaring and initialization one, and two-dimensional arrays with suitable examples.	6	L2	CO3
	c	Develop a program to read N numbers and find the sum and average of N numbers using an array.	8	L3	CO3
Module-4					
Q. 07	a	Define function in C. Justify the need of user defined functions in C with a suitable program.	8	L2	CO4
	b	Explain with example "No arguments and no return values" of functions.	6	L2	CO4

Model Question Paper- I

	c	Develop a C program to perform arithmetic operations (+, -, /, *) using user defined functions.	6	L3	CO4
OR					
Q. 08	a	List and Explain the various elements of user defined functions.	8	L2	CO4
	b	What is nested function? Explain with example.	6	L2	CO4
	c	Develop a suitable program having a function with arguments and no return value.	6	L3	CO4
Module-5					
Q. 09	a	Define structure. Explain the general format of a structure definition.	6	L2	CO5
	b	Define pointer. Illustrate declaring and initialization of a pointer variable.	6	L2	CO5
	c	Define a structure type student that would contain student name, 3 subject marks Using this structure, Develop a C program to read four students data from keyboard and print the same on the screen.	8	L3	CO5
OR					
Q. 10	a	Differentiate between arrays and structures with an example.	6	L2	CO5
	b	Illustrate with suitable code for swapping of two numbers using pointers.	6	L2	CO5
	c	Develop a program to copy and compare of structure variables.	8	L3	CO5

AY: 2025-26

Department: Computer Science and Engineering
Subject with Sub. Code: Introduction to C Programming(1BPLC205E)
Sem/Div: II/ A&B
Name of Faculty: Prof. Shree Gowri S S

Q.No.	Solution and Scheme	Marks
	Module - 1	
1a.	Defn of Algorithm:- An algorithm is a finite set of unambiguous instructions which when executed, performs a task correctly It has three characteristic features 1. No of steps required to perform the task should be finite 2. Each of the instrn in the algorithm should be unambiguous in nature 3. finally, algorithm should solve the problem correctly	03M
	Algorithm to convert temperature from Fahrenheit to Celsius Step : 1. Start 2. Input temperature in Fahrenheit 3. calculate Celsius using formula $C = (F - 32) \times \frac{5}{9}$ 4. stop	05M
1b.	functions :- A function is a named block of code that performs a specific task. A function allows to write a piece of logic once & reuse it wherever & whenever needed in the program. It helps to keep code in organized, easier to understant & manage	01

Q.No.	Solution and Scheme	Marks
	Structure of C-program	04 / 05M

* Documentation Section
* Link Section
* Definition Section
* Global Declaration Section

main () function section
{
  Declaration part
  Executable part
}
Subprogram Section
  function1
  function2
  functionn

fig: an overview of C-program

* Every C-program has one main() function that contains two parts
  1) declaration part
  2) Executable part

Q1C Input & Output functions

Reading a character : done by using getchar();

Syntax:- variable_name = getchar();

Ex:- char name;
name = getchar();

Writing a character :- putchar is used for writing character one at a time

Syntax:- putchar (variable-name);

Ex:- answer = 'Y';
putchar (answer);

formatted Input

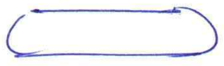
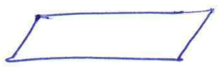
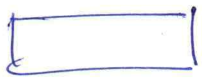
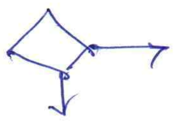
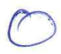
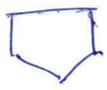
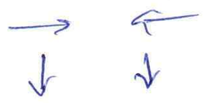
scanf ("control string", arg1, arg2, ... argn);

Inputting Integer Number

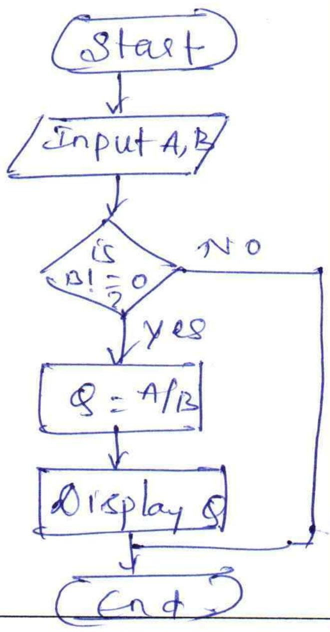
Ex:- scanf ("%2d %5d", &num1, &num2);

Q.No.	Solution and Scheme	Marks
	Inputting real numbers Scanf ("%f %f %f", &x, &y, &z);	07
	OR	
2a.	Flow chart is a tool to represent logic. It helps to graphically visualize the flow control within the sequence of statement.	08

Symbols used in flow chart

Name	Symbol
Terminal	
Input/output	
Processbox	
Decision box	
Connector	
Off-page Connector	
Arrowed lines	

Example



Q.No.	Solution and Scheme	Marks
2b	Features of C-programming. 1. C - is a robust language whose rich-set of built-in functions & operators can be used to write any complex program. 2. Program written in C are efficient & fast 3. It has 32 keyword & built-in function 4. It is high-level language & that is suitable for writing both system software & business packages. Compiling and Executing a C-program 1. Creating the program 2. Compiling the program 3. Linking the program with functions that are needed from the C-library 4. Executing the program	05
2c.	Identifier:- Every C-word is classified as either a keyword or an identifier. All keywords have fixed meaning & these meanings cannot be changed. Identifier is a name given to variable or user defined function Rules for identifier 1. first character must be alphabet 2. Must consist of only letters, digits or underscore 3. Only first 31 characters are significant 4. Cannot use keyword 5. Must not contain white space Ex:- Total, abc, abc_123 etc	07

Q.No.	Solution and Scheme	Marks
	Module-2	

8a.	<u>operators</u>
	1) Increment & Decrement operator
	<u>Increment operator</u> ++
	Ex :- ++m or m++;
	++m; is equivalent to m=m+1;
	or m+=1;
	<u>Decrement operator</u> --
	Ex :- --m or m--;
	--m; is equivalent to m=m-1;
	or m-=1;
	Ex :- m=5
	y=++m; then y=6;
	<u>Logical operator</u>
	C-programming uses logical operator to perform logical operations
	<u>operator</u> <u>Meaning</u>
	&& Logical AND
	|| Logical OR
	! Logical NOT
	Ex :- 1) a > b && x == 10
	2) if (age>55 && salary<1000)
		08

8b.	<u>else-if ladder</u>
	Syntax:- if (condition 1)
	Statement 1;
	else if (condition 2)
	Statement 2;
	else if (condition 3)
	Statement 3;
		06

Q.No.	Solution and Scheme	Marks

	else if (condition n)
	    statement n;
	
	else
	    default statement;
	
	Statement x;
	
	else-if ladder :- A multipath decision is a chain of ifs in which the statement associated with each else is an if.
	
	Ex :- if (marks > 79)
	    grade = 'H'
	else if (marks > 59)
	    grade = '1';
	else if (marks > 49)
	    grade = '2';
	else if (marks > 39)
	    grade = '3';
	else
	    grade = 'F';
	printf ("%c \n", grade);	

8c	Syntax for switch statement
	switch (expression)
	{
	  case constant1:
	      // statements;
	      break;
	
	  case constant2:
	      // statements;
	      break;
	  - -
	  default:
	      // statements;
	  break;
	}	

Q.No.	Solution and Scheme	Marks

Program

```
# include <stdio.h>
int main()
{
   int day;
   printf ("Enter a number (1-7): ");
   scanf ("%d", & day);
   Switch (day)
   {  case 1 :
            printf ("Sunday");
            break;
      case 2 :
            printf ("Monday");
            break;
      case 3 :
            printf ("Tuesday");
            break;
         .
         .
         .
      case 7 :
            printf ("saturday");
            break;
      default:
            printf (" Invalid Input ! please
                     enter number between
                       1 and 7 ");
      }
      return 0;
   }
```

OR

Q.No.	Solution and Scheme	Marks
4a	Evaluation order of the following expression $100/20 <= 10-5+100\%10-20==5>!!=20$ Step1 :- $100/20 = 5$ Division & Modulus operation $100\%10 = 0$	08

Q.No.	Solution and Scheme	Marks

$5 <= 10 - 5 + 0 - 20 == 5 >= 1 != 20$

<u>Step2</u> : Addition & subtraction

$10 - 5 = 5$
$5 + 0 = 5$
$5 - 20 = -15$

∴ $5 <= -15 == 5 >= 1 != 20$

Step 3 : Relational operators $(<, <=, >, >=)$

1. $5 <= -15 \rightarrow$ false (0)
2. $5 >= 1 \rightarrow$ True (1)

∴ $0 == 1 != 20$

Step 4 : Equality operator $(==, !=)$

1 $0 == 1$ false (0)
2. $0 != 20$ True (1)

∴ final answer = 1 (True)

4b. Entry controlled loop & exit controlled loop — 06

<u>Entry control</u>	<u>Exit control loop</u>
1. condition is checked at the entry of loop	1. condition is checked at the exit of the loop
2. If the condition is false then loop will not execute	2. At any condition loop will execute atleast once
3. Ex :- while loop, for loop	3. Ex :- Do-while loop

4c. C-program to find sum of first n numbers using while loop — 06

Q.No.	Solution and Scheme	Marks

Program

```
# include <stdio.h>
int main () {
   int n, sum = 0, count = 1;
   printf (" Enter a positive integer n: ");
   if (scanf ("%d", &n) != 1 || n <= 0) {
    printf (" Invalid input. Please enter valid
                     positive integer \n");
    return 1;
   }
   while  (count <= n) {
          Sum += count;
          Count ++;
   }
   printf (" The sum of first %d numbers is
                        : %d \n", n, sum);
   return 0;
}
```

Module 3

5 a. String is a Sequence of characters terminated with a null character \0

Program

```
# include <stdio.h>
# include <string.h>
int main()
{ char str [100];
  int i, length;
  printf (" Enter a String: ");
  scanf ("%s ", str);
  length = strlen (str),
  printf (" Reversed string: ");
```

Q.No.	Solution and Scheme	Marks

```
for (i= length -1; i>=0; i--)
{ printf ("%c", str[i]);
}return 0;
}
```

5b. Array:- array is a fixed size sequenced collection of elements of the same data type. In its simplest form, an array can be used to represent a list of numbers or list of names 06 [2+4]

Ex:- salary [10]

Types of Array

1) One-dimensional array
2) Two-dimensional array.

1) One-Dimensional Array

General form

type variable-name[size];

Ex:- float height [50];

initialization of one-dimensional array

Array can be initialized at either of the following stage

1) At complite time 2) At run time.

Compile Time initialization

General form

type array_name[size]= {list of values};

Ex:- 1) int number [3] = {0,0,0};

2) float total[5] = {0.0, 15.75, -10,4}

Two-Dimensional array

Syntax:- type array-name [row-size][column-size]

Ex:- v[4][3]

Q.No.	Solution and Scheme	Marks
	initializing Two - Dimensional array Ex :- int table [2][3] = {0, 0, 0, 1, 1, 1};	
5c.	program to multiply two N*N matrices	08

```
#include <stdio.h>
int main () {
 int n, i, j, k;
 printf (" Enter the order of matrix (n)": );
 Scanf ("%d", &n);
 int A[n][n], B[n][n], c[n][n];
 printf (" Enter elements of first matrix : \n");
 for (i=0; i<n ; i++) {
       for ( j=0; j<n; j++) {
             Scanf ("%d", &A[i][j]);
       }
  }
 printf (" Enter elements of Second matrix : \n");
 for (i = 0; i<n ; i++) {
       for (j=0; j<n; j++) {
          Scanf (" %d", &B[i][j]);
       }
   }
 for (i=0; i<n; i++) {
       for [j =0 ; j<n ; j++) {
             C[i][j] = 0;
       }
   }
 for (i=0; i<n; i++){
       for (j=0 ; j<n; j++) {
             for (k=0 ; k<n ; k++) {
                  C[i][j] += A[i][k] * B[k][j];
       }
  }}
```

Q.No.	Solution and Scheme	Marks

```
printf (" Result matrix :\n");
 for (i=0;  i<n;  i++) {
      for(j=0; j<n; j++) {
          printf(" %d",  c[i][j]);
   }
  printf ("\n");
 }
  return0;
 }
```

OR

6a. Program to find length of string without using built in function — 06

```
# include <stdio.h>
 int main() {
   char str[100];
    int  i=0, length =0;
   printf(" Enter a string : ");
   fgets (str, sizeof(str), stdin):
    while ( str [i] != '\0') {
      if  (str[i]! = '\n') {
              length++;
        }
         i++;
    }
     printf(" length of the string = %d\n",
                                      length);
     return 0;
   }
```

Q.No.	Solution and Scheme	Marks

6b. Declaration of one-dimensional array

Syntax: type variable_name [size];

Ex:- int hight [50];

Initialization of one-dimensional array

Initialization can be done at

i) Compilation time or Run time

Compilation Time initialization

Syntax:-

```
type array_name [size] = {list of values};
```

Ex:- int number [5] = {0, 0, 0, 0, 0};

Run-time initialization

Sy. Ex:-

```
for (i = 0; i<100; i = i+1)
{
    if i< 50
        sum [i] = 0.0;
    else
        sum [i] = 1.0;
}
```

Two-Dimensional Array

Syntax:-

```
type array_name [row size] [column size]
```

Ex:- int v [4] [3]

Initializing Two-Dimensional Array

```
int table [2] [3] = {0, 0, 0, 1, 1, 1};
```

or

```
int table [2] [3] = {
                      {0,0,0};
                      {1,1,1};
                    };
```

Q.No.	Solution and Scheme	Marks

6 c.	Program to read N numbers & find the sum and average of N numbers using an array
	`# include <stdio.h>`
	`int main() {`
	`int n, i;`
	`float sum=0, average;`
	`printf(" Enter number of elements: ");`
	`scanf("%d", &n);`
	`float arr[n];`
	`printf(" Enter %d numbers:\n", n);`
	`for (i=0; i<n; i++) {`
	`scanf("%f", &arr[i]);`
	`}`
	`for (i=0; i<n; i++) {`
	`sum += arr[i];`
	`}`
	`average = sum/n;`
	`printf(" Sum = %.2f\n", sum);`
	`printf("Average = %.2f\n", average);`
	`return 0;`
	`}`
	
	<u>Module-4</u>
		8 -

7a.	<u>Defn of function in C</u> :- Function in C is a block of code that performs a specific task
	<u>Need for user-defined function</u>
	`#include <stdio.h>`
	`int square (int num){`
	`return num*num;`
	`}`
		8M

Q.No.	Solution and Scheme	Marks

```
int main () {
  int number, result;
  printf ("Enter a number: ");
  scanf ("%d", &number);
  result = Square (number);
  printf ("Square of %d = %d", number,
                              result);

  return 0;
 }
```

Justification

- Square() is a user - defined function
- It takes one argument num
- It returns the square of a number
- main() calls the function & prints the result.

7b. "No arguments & no return values" 06

when a function has no arguments, it does not receive any data from calling function

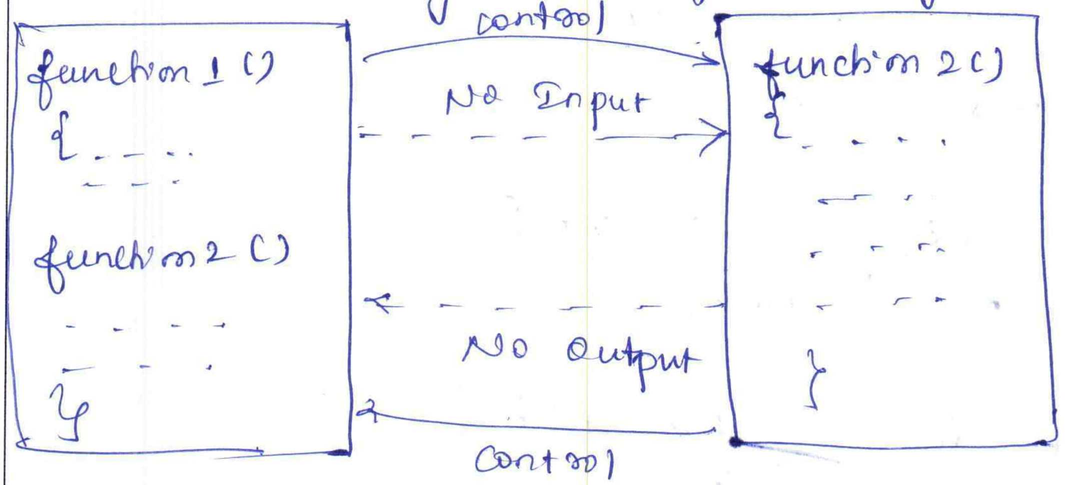


fig: No data Communication between function.

when it does not return a value, the calling function does not receive any data from called function. In effect, there is no data transfer between calling & called function

Q.No.	Solution and Scheme	Marks
	Arguments with No return value	

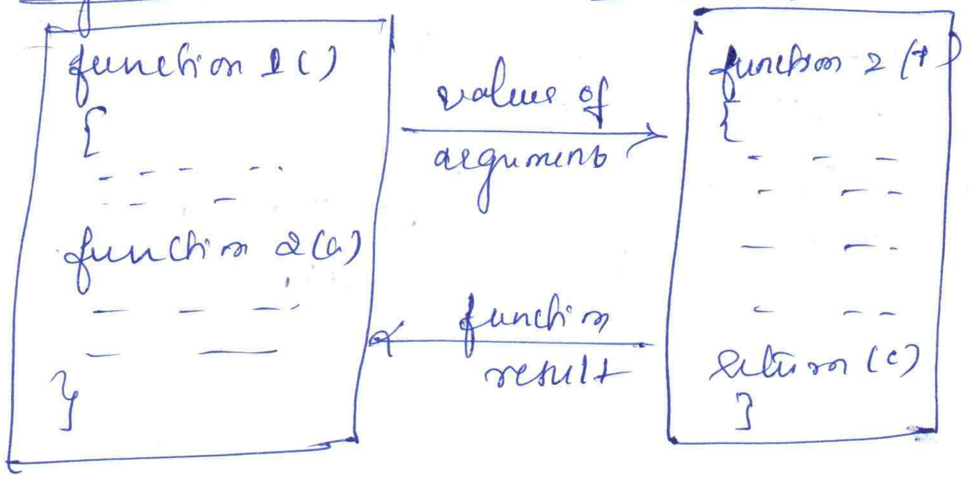


fig: Two way Data Commⁿ between function

~~4~~ C. Program to perform arithmetic operations(+, -, *, /) 08m
using user defined functions

```
# include <stdio.h>
 int add (int a, int b) {
      return a+b;
 }
 int Sub ( int a, int b) {
      return a-b;
 }
 int multiply (int a, int b){
       return a*b;
 }
 float divide (int a, int b) {
        return a/b;
 }
int main () {
   int x, y;
   printf(" Enter two Numbers: ");
   Scanf (" %d %d", &x, &y);
```

Q.No.	Solution and Scheme	Marks

```
printf ("Addition = %d \n", add (x, y));
printf ("Subtraction = %d \n", sub (x, y));
printf ("Multiplication = %d ~~ef~~ \n", multiply(x,y));
~~print~~ if (y != 0)
    printf ("Division = %.2f \n", divide(x,y));
    else
      printf ("Division by zero is not allowed");
      return 0;

}
```

<u>OR</u>

8a. Various elements of user-defined function are :- **08**

1. function definition
2. function call
3. function declaration

Function defn :- It is independent program module that is specially written to implement the requirements of the function.

The function can be invoked when & where its required and it is <u>function call</u>

The program (function) that calls a function is referred to as <u>calling function</u>

The calling program should declare any function and it is known as <u>Function declaration</u>

Q.No.	Solution and Scheme	Marks

	Function implimentation shall include the following elements
	1) function name 4) Local variable declaration
	2) function type 5) function statement
	3) list of parameters 6) return statements
	
	All these six elements are grouped together in two parts namely
	1) function header (first 3 elements)
	2) function body (2nd 3 elements)
	General format
	function_type function_name (parameter list)
	{
	local variable declaration;
	executable statement 1;
	executable statement 2;
	- - -
	- -
	return statement;
	}	

8b.	Nested function with example C permits nesting a function freely. main call function 1, which call function 2, which calls function 3, - - - & so on. There is no limit how deeply function can be nested example:- float ratio (int x, int y, int z); int difference (int x, int y); main() {	06

Q.No.	Solution and Scheme	Marks
	int a, b, c; scanf ("%d %d %d", &a, &b, &c); printf ("%f \n", ratio(a, b, c)); } float ratio (int x, int y, int z) { if (difference (y, z)) return (x/ (y-z)); else return (0.0); } int difference (int p, int q) { if (P! = q) return (1); else return (0); }	
	The above program calculates the ratio $\frac{a}{b-c}$ where in the program contains the three function main(), ratio(), differen()	
8c.	C - program having a function with argument and no return value	06 M
	#include <stdio.h> void display (int num) { printf (" Entered number is : %d \n", num); } int main() { int number; printf (" Enter a number : "); scanf ("%d", & number); display (number); return 0; }	

```
int a, b, c;
scanf ("%d %d %d", &a, &b, &c);
 printf ("%f \n", ratio(a, b, c));
 }
 float ratio (int x, int y, int z)
 {   if (difference (y, z))
       return (x/ (y-z));
       else
         return (0.0);
   }
    int difference (int p, int q)
    {
       if (P! = q)
           return (1);
       else
         return (0);
    }
```

The above program calculates the ratio $\frac{a}{b-c}$
where in the program contains the three
function main(), ratio(), differen()

8c. C - program having a function with argument and no return value — 06 M

```
#include <stdio.h>
 void display (int num) {
  printf (" Entered number is : %d \n", num);
 }
 int main() {
   int number;
   printf (" Enter a number : ");
   scanf ("%d", & number);
   display ( number);
   return 0;
 }
```

Q.No.	Solution and Scheme	Marks
	Module 5	

9a.	Defn of structure :- Structure is a convenient tool for handling a group of logically related data items
	Ex :- It can be used to represent set of attributes Ex :- Student_name, roll_no, marks
	
	General format of Structure
	struct tag_name
	{
	data_type number1;
	data_type number2;
	- - - - - -
	- - - - - -
	}
		06

9b.	Pointer :- A pointer is a derived data type in c. It is built from one of the fund-amental data types available in C. Declaring a pointer variable Syntax :- data type *pt_name; Ex :- int *p; float *x; initializing pointer variable once pointer variable is declared, assignment operator can be used to initialize the variable	(1+5) 06

Q.No.	Solution and Scheme	Marks

	Ex:- int quantity
	int *p;
	p = & quantity;
	we can also combine initialization with declaration
	int *p = & quantity;	

9c.	Program to define structure type for student	08

```
# include <stdio.h>
 struct student {
   char  name[50];
   int   marks1, marks2, marks3;
 };
 int main() {
   struct student  s[4];
   int i;
   for (i=0; i<4; i++) {
   printf("\Enter details of student %d\n",
                                    i+1),
   printf ("Name: ");
   scanf ("%s", s[i].name);
   printf ("Marks in 3 Subjects: ");
   scanf ("%d %d %d" , &s[i].marks1,
          & s[i].marks2, &s[i].marks3);
   printf ("\nStudent Details :\n");,
   for (i=0; i<4; i++) {
   printf(" \nStudent %d\n", i+1);,
   printf(" Name: %s \n", s[i].name);
   printf (" Marks: %d %d %d \n");
```

Q.No.	Solution and Scheme	Marks

	s[i], marks 1,
	s[i]. marks 2,
	s[i]. marks 3);
	} return 0;
	}
	
	OR	

10 a.	Difference between array & structure Array: 1) Stores same type of data — Structure: 1. Stores different type of data Array: 2) Elements are accessed using index — Structure: 2. Members are accessed using dot (.) operator Array: 3) Declared using data type — Structure: 3. Declared using struct keyword Array: Ex:- List of Marks — Structure: Ex:- Student record	06
10 b.	program to swap two numbers using pointer	06

```
# include <stdio.h>
void swap (int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
 int main () {
  int x, int y;
  printf (" Enter two Numbers: ");
  Scanf ("%d %d", &x, &y);
  printf (" Before swapping : x = %d, y = %d \n",
                                              x, y);
  swap (&x, &y);
  printf (" After swapping x = %d , y = %d \n",
                                              x, y);
 return 0;
}
```

Q.No.	Solution and Scheme	Marks
10 C.	Program to copy & compare of structure variable	8

```
#include <stdio.h>
#include <string.h>
struct student{
    int roll;
    char name[50];
    float marks;
  };
  int main() {
  printf(" Enter roll number: ");
  scanf("%d", &s1.roll);
  printf(" Enter name: ")
  scanf("%s", & s1.name);
  print(" enter marks: ");
  scanf("%f", &s1.marks);
  s2 = s1;
  printf("\n... After copying ...\n");
  printf("Roll : %d \n", s2.roll);
  printf("Name: %s \n", s2.name),
  printf(" Marks : %.2f \n", s2.marks);
  if (s1.roll == s2.roll &&
      strcmp (s1.name , s2.name) == 0&&
      s1.marks == s2.marks) {
      printf("\n structures are equal \n");
    }
    else
     {
      printf(" structures are not equal\n");
     } return 0;
   }
```

[illegible] [illegible] [illegible]