Code No.: R25CS104ES | R25 | H.T.No. | | | 8 | R | | | | | |

CMR ENGINEERING COLLEGE: : HYDERABAD
UGC AUTONOMOUS
I–B.TECH–I–Semester End Examinations (Regular) - December - 2025
PROGRAMMING FOR PROBLEM SOLVING
(Common for ECE, CSE, CSM & CSD)

[Time: 3 Hours] **[Max. Marks: 60]**

Note: This question paper contains two parts A and B.
Part A is compulsory which carries 10 marks. Answer all questions in Part A.
Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

PART-A **(10 Marks)**

1. a)	Explain about printf() and scanf() .	[2M]
b)	What is actual parameter and formal parameter?	[2M]
c)	Difference between gets () and puts ().	[2M]
d)	What is recursive function?	[2M]
e)	How linear search differentiated from binary search?	[2M]

PART-B **(50 Marks)**

2.a)	Explain the structure of C program.	[5M]
b)	Differentiate between while and do-while using an example.	[5M]
	OR	
3.	What is conditional statement? List Various Conditional Statements used in C. (if, if-else, nested if-else, switch).	[10M]
4.	Explain in detail about the function without argument and without return type with an example program.	[10M]
	OR	
5.	Explain various storage classes in c program.	[10M]
6.	What is an array? Write a program to implement addition of two matrices using 2D arrays.	[10M]
	OR	
7.	What is string? Explain basic string handling functions with an example.	[10M]
8.	What is Structure? Explain how to access members of structure using C program.	[10M]
	OR	
9.a)	What is a pointer? How pointer is declared and initialized.	[5M]
b)	Differentiate between structure and union.	[5M]
10.	What is sorting? Explain Bubble sort with an example.	[10M]
	OR	
11.	Write an algorithm to implement linear search in an array with example.	[10M]

****

CMR ENGINEERING COLLEGE

I B.Tech. I Sem (Regular) DEC-2025

Programming Problem Solving (R25CS104ES)

SCHEME

PART - A

1a) Explain about printf() and scanf()

printf() - output function

It prints text and variable values to the console using a format string with format specifiers like %d, %f ...

printf("Format String", argument_list);

Eg: int a=10;
printf("Value of a = %d", a);

scanf(): Input function

It reads data from the user and stored in the variable.

scanf("Format specifiers", &var1, &var2);

Eg: int x;
printf("Enter a number");
scanf("%d", &x);

b) What is actual parameter and Formal parameter

Actual Parameter: These appear in the function call and are the real values, variables or expressions given to the function

Eg: sum(5, x); here 5 & x are actual parameters

Formal Parameter: These are the variables declared in the function header/definition that receive the actual values.

Eg: int sum(int a, int b); → a & b are Formal Parameters

c) Difference between gets() and puts()

gets(): It reads a line of text from the user & stores it into a characters array. It is unsafe because it does not check the array size, which can cause buffer overflow.

puts(): outputs a null-terminated string to the screen and automatically adds a newline at the end of the printed string.

d) What is recursive function

A function call itself directly or indirectly.

```
void add()
{
  ...
  add();
  ...
}
```

e) How linear search differentiated from binary search?

Linear search	Binary search
- Works on sorted or unsorted arrays	- Requires the array to be sorted.
- Sequential comparison of each element with the key	- Divide-and-Conquer, compare with middle, then search left or right half.
- Simpler to understand and code	- More complex than linear search.

PART-B

2 a) Explain the structure of C program.

Sol: Basic structure of a C program is

- Documentation section
- Link section
- Definition section
- Global declaration section
- main() Function section

```
{
    Declaration part
    Executable part
}
```

- Subprogram section or user defined functions

- **Document section** : It consist of a set of comment lines giving the name of the program & other details like author, date written, date compiled etc.

  Single line comments : written after //

  Multiline comment : written between /* and */

- **Link section or File Inclusion section:**

  It provides instructions to the compiler to link object code of the functions from the system library. Each header file has extension '.h'. The header files are included at the beginning of the program. These files should be included using #include directive

  Eg. #include <stdio.h>
  #include <stdlib.h>

- Definition section :

It defines all symbolic constants and macros used in the program. Preprocessor is the system program which replaces all the symbolic constants with the values before compilation of the program. #define is used for defining a constant or macro.

Ex #define PI 3.14

#define cub(x) x * x * x

- Global Declaration section :

Global variables can be shared by all the functions in the c program. -

- Main() Function section :

Every c program must have one main() function section. All programs execution start from here.

→ Sub program section :

This section contain definitions of user-defined functions. these are used to perform specific tasks within your program.

2b) Differentiate between while and do-while using an example.

Sol,

While Loop

- It is also called as Entry-Controlled loop
- Here, the condition is checked before the execution of the loop body
- Syntax:

```
while (condion)
{
    :: // body of the loop
    Increment/Decrement
}
```

- Here we may or may not get the output
- w.a.p Print natural numbers up to 15

```
#include <stdio.h>
void main()
{
    int n=15;
    int i=1;
    while (i<=n)
    {
        printf("%d\t", i);
        i++;
    }
    printf("\n");
}
```

do-while Loop

- It is also called as Exit-Controlled loop
- Here, the condition is checked after the execution of the loop body
- Syntax:

```
do
{
    -- // body of the loop
    Increment/Decrement
}while(condition);
```

- Here, we get minimum one time output.
- w.a.p to Print natural numbers up to 15

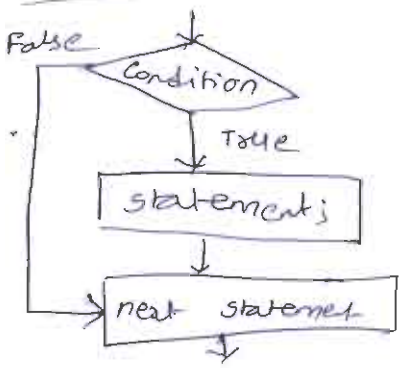
```
#include<stdio.h>
void main()
{
    int i, n;
    i=1;
    n=15;
    do
    {
        printf("%d\t", i);
        i++;
    }while(i<=n);
    printf("\n");
}
```

3) What is Conditional statement? List various conditional statements used in C?

Sol: - if statement

It is used to execute set of statements if the condition is true. It works only when condition is true, it does not specify what to do if the condition is false.

flow chart:



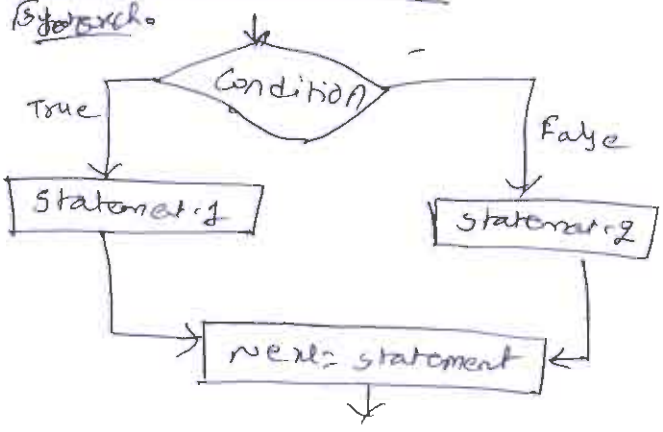
Syntax:

```
if(expression)
{
    statements;
}
next_statement;
```

Eg: #include <stdio.h>
void main()
{ int age = 19;
  if (age >= 18)
  {
    printf("Eligible for voting");
  }
}

- if-else statement:

flowchart:



Syntax:

```
if (condition)
{   statement_1;
}
else
{   statement_2;
}
Next_statement;
```

Execution: The Condition is evaluated first if it is true the "Statement-1" block executed and Control transfer to "next statement". If the Condition is false then "statement-2" block is executed and Control transfer to "next-statement".

Example Program:

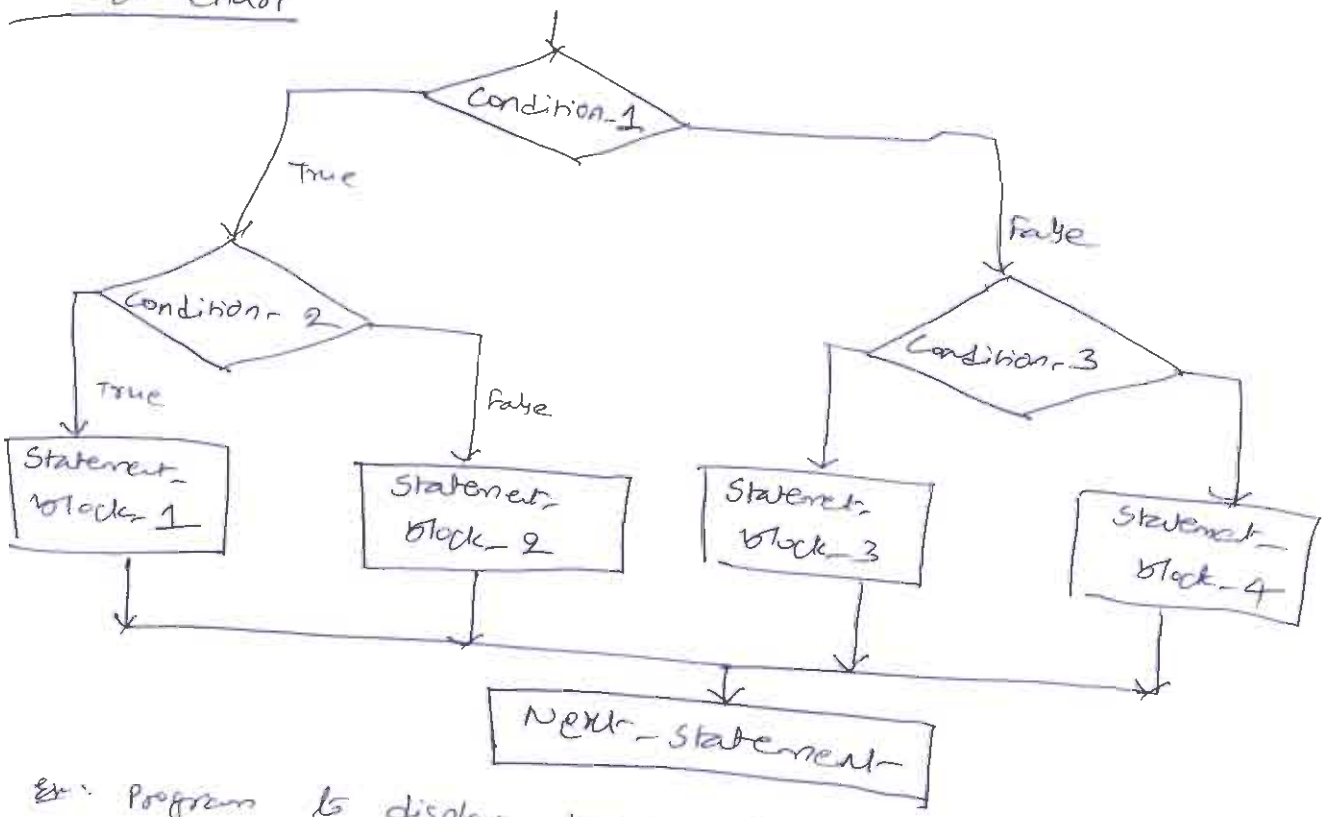
```
#include <stdio.h>
void main()
{ int n;
  printf("Enter a number:");
  scanf("%d", &n);
  if (n%2 == 0)
  {
     printf("%d is even number.", n);
  }
  else
  {
     printf("%d is odd number.", n);
  }
}
```

- Nested if else statement-

syntax:

```
if (condition-1)
{
    if (condition-2)
    {
       statement block-1;
    }
    else
    {
       statement block-2;
    }
}
else
{
    if (condition-3)
    { statement block-3;
    }
    else {
```

Flow chart



Ex: Program to display largest of three numbers

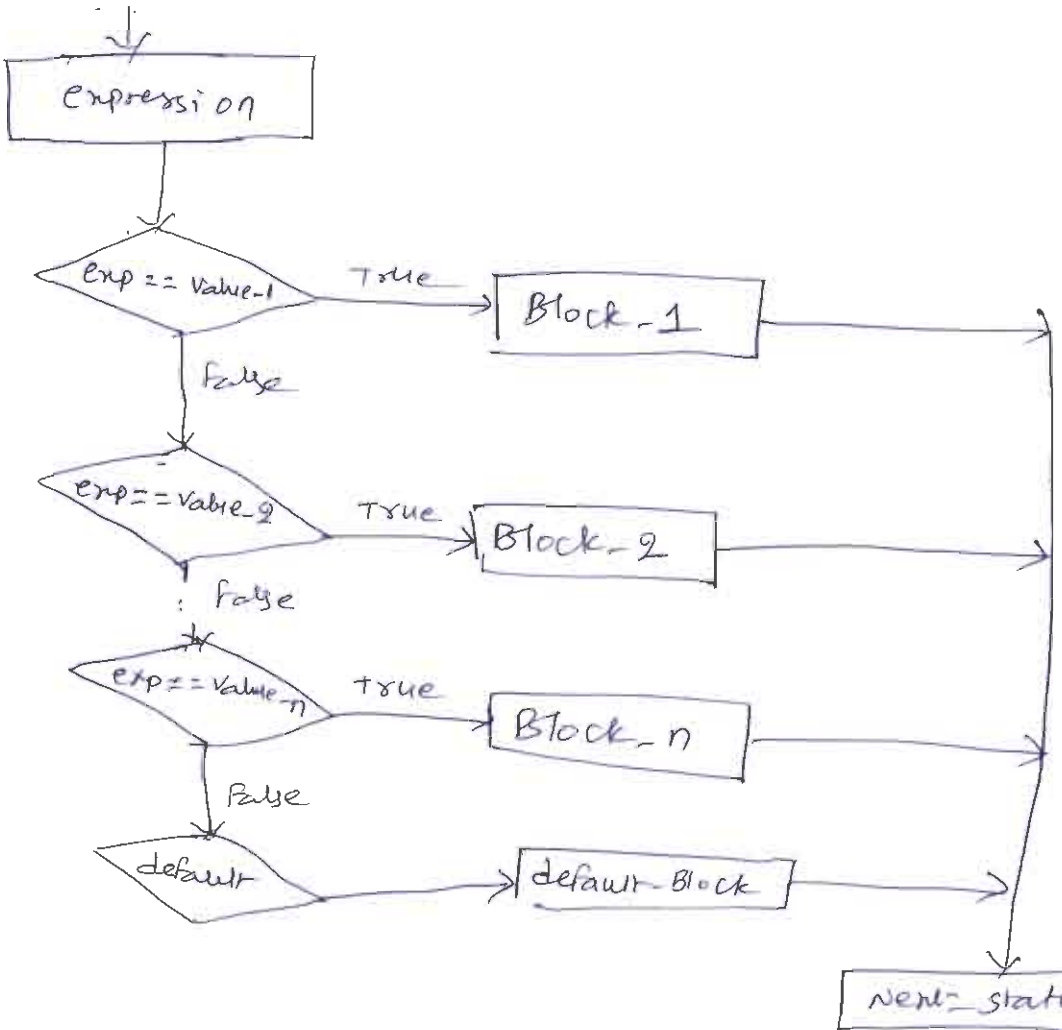
```
#include
void main()
{ int n1, n2, n3; l;
  printf("Enter Three Numbers: ");
  scanf("%d %d %d", &n1, &n2, &n3);
  if((n1 >= n2))&& (n1 >=
  {
      if (n1 >= n3)
          l = n1;
      else
          l = n3;
  }
  else
  {
      if(n2 >= n3)
      {
          if(n2 l = n2;
      }
      else
      {
          l = n3;
      }
  }
  Printf("%d is largest number.", l);
}
```

→ switch statement :

Syntax :

```
switch ( expression )
{
    case value_1 :
                    Block_1;
                    break;
    case value_2 :
                    Block_2;
                    break;
    .
    .
    .
    case value_n :  Block_n;
                    break;
    default      :
                    default_Block;
                    break;
}
next_statement;
```

Flow_chart :



Execution

Here, expression is an integer / character constant.

- expression is checked with values. When the exp == value-x then corresponding block is executed and control goes to next-statement.
- If all values are mismatches then default-block is executed and control goes to next-statement.

Program: Program to display whether a given number is even or odd.

```
#include <stdio.h>
void main()
{
    int n;
    Printf("Enter a number:");
    scanf("%d", &n);
    switch (n%2)
    {
        case 0: Printf("%d is even.", n);
                break;
        case 1;
                Printf("%d is odd", n);
                break;
        default:
                Printf("wrong number");
    }
}
```

Q4: Explain in detail about the Function without argument and without return type with example Program.

Definition: of Function: A Function is an independent module that will be called to do a specific task. (or)

A Function is a self-contained block that carries out a specific well defined task.

Syntax:

```
return_type  Function_name (arguments / Parameters list)
{
    local Variable declaration;
    statement 1;
    statement 2;
    return (value);
}
```

→ Types of Functions with Respect to arguments and return-type

1. Function without arguments and return values.
2. Function with arguments but without return values
3. Function with arguments and return values
4. Function without arguments and but with return values

⇒ Function without arguments and without return type.

```
Calling Function                    Called Function
void main()                         void abc()
{                                   {
  ...                                 ...
}                                   }
void abc();
...
}
```

Analysis: No arguments are passed. No values are sent back

- Here, Data is neither passed through the calling function nor sent back from the called Function.

- There is no data transfer between calling and called functions
- The function is only executed and nothing is obtained.
- If such functions are used to perform any operation they act independently. They read data values and print result in the same block.
- Such functions may be useful to print some message, draw a line or split the line.

Program

```
#include <stdio.h>
void message( )
void main( )
{
   message( );
}
void message( )
{
   Printf (" Have a nice day ");
}
```

Output

Have a nice day.

Q 5) Explain Various storage classes in C program.

Storage classes in C are used to determine the lifetime, visibility, memory location and initial value of a variable. There are 4 types of storage classes in C.

1. Automatic storage class

- For this variables memory will be allocated automatically at runtime
- The visibility of the automatic variables is limited to the block in which they are defined.
- The initial value is garbage
- The memory assigned to automatic variables gets freed upon exiting from the block.
- The keyword used for defining automatic variable is auto
- Every local variable is automatic by default.

Program:

```
#include<stdio.h>
void main()
{
    auto int x = 5;
    Printf(" x = %d", x);
}
```

2. Static storage class

- These variables hold their value between the multiple function calls
- Default initial value is 0 or null
- static local variables are visible only to the function or block in which they are defined.

- The visibility of the static global variable is limited to the file in which they are defined.
- keyword used to define static variable is "static"

Program:

```
#include <stdio.h>
void increment( )
{
    static int x=0;
    x++;
    Printf("%d\n", x);
}
void main( )
{
    increment( );
    increment( );
    increment( );
}
```

Output

```
1
2
3
```

- Static variables are used in Functions that have a local scope but retain their value between function calls.

3. Register Storage class

- These variables are declared with keyword "register"
- These variables are stored in CPU registers depending upon the size of the memory remaining in the CPU.
- We can't use & operators.
- The default initial value is '0'

- we can store pointer into the register.
- The scope local to the block or function
- The lifetime is within the function

Program:

```
#include<stdio.h>
int main()
{
    register int x = 5;
    printf("x = %d", x);
    return(0);
}
```

Output

x = 5

NOTE: Generally register variables execution speed is very fast. The loop control variables are stored with "register" storage class.

4. External storage class

- These variables are declared with the keyword "extern"
- The default initial value is "zero".
- we can only initialize the extern variable globally.
- An extern variable can be declared many times but can be initialized at only once.
- The scope of these variables is ~~Local~~ Global to ~~Static~~ Program/file
- The lifetime of extern variable is Till the end of the main program. Maybe declared anywhere in the program.

Program:

File2.c

```
#include <stdio.h>
#include "File1.c"
extern x;
extern void print_x();
void main()
{
   printf("x = %d\n", x);
   print_x();
}
```

File1.c

```
#include <stdio.h>
int x = 5;
void print_x()
{
   printf("x = %d", x);
}
```

OUTPUT

```
x = 5
x = 5
```

Q.no 6) What is an array? Write a program to implement addition of two matrices using 2D arrays?

Sol:- Array: Array is nothing but collection of similar data elements that share single/common name.

- All the elements in the array stored in continuous memory location.

- Declaration:

```
data-type  variable_name [size];
```

eg: int a[10];

Program:

```
#include <stdio.h>
void main()
{ int a[10][10], b[10][10], c[10][10], i, j, r1, c1, r2, c2;
  printf("Enter row & column size of matrix-1");
  scanf("%d %d", &r1, &c1);
  printf("Enter row & column size of matrix-2");
  scanf("%d %d", &r2, &c2);
  if((r1==r2) && (c1==c2))
  {
    printf("Enter elements into matrix-1");
    for(i=0; i<r1; i++)
      for(j=0; j<c1; j++)
        scanf("%d", &a[i][j]);
    printf("Enter elements into matrix-2");
```

```
for(i=0; i<r2; i++)
    for(j=0; j<c2; j++)
        scanf("%d", &b[i][j]);

for(i=0; i<r1; i++)
    for(j=0; j<c1; j++)
        c[i][j] = a[i][j] + b[i][j];

printf("Addition of Matrices is \n");
for(i=0; i<r1; i++)
{
    for(j=0; j<c1; j++)
    {
        printf("%d \t", c[i][j]);
    }
    printf("\n");
}
}
```

Q.7) What is string? Explain basic string handling function with example.

Sol- - The string is a collection of characters enclosed in double quotes (" ") and ends with NULL character. ('\0')

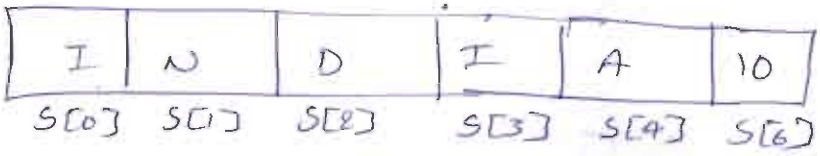
- String is stored in an array of characters.
- The string is a derived datatype.

Declaration Syntax:

```
char string_var_name [size];
```

Eg: `char s[6];`

Initialization: `s[6] = {'I', 'N', 'D', 'I', 'A', '\0'};`



- In string, the last character is always '\0'. It is not compulsory to write '\0' in string. The compiler automatically puts '\0' at the end of the string.

⇒ Some string handling Functions

- All string handling functions are available in "string.h" header file.

1. strlen()

This function counts the number of characters in a given string.

Syntax: `strlen(char *s);`

Eg:
```
char text = "Hello";
int len = strlen(text);
```

O/P:
```
len = 5
```

2. strcpy()

This function copies the contents of one string into another

Syntax:

```
strcpy (char *s2, char *s1);
```

where s1 is copied to s2.

Eg.. char ori[20] = "sachin"; char dup[20];

strcpy (dup, ori);

O/P:

ori = sachin

dup = sachin

3. strcmp():

Syntax:

```
strcmp (char *s1, char *s2);
```

The above function compares two strings for finding whether they are the same or different. Characters of these strings are compared one by one. In case of a mismatch the function returns 0 or non-zero value, otherwise it return zero. If they are different it returns the numeric difference between the ASCII value of non-matching characters

Ex:

```
char str1[20] = "AB";
char str2[20] = "Ab";
int x;
x = strcmp (str1, str2);
```

O/P:

x = -32;

4. strlwr()

This function can be used to convert any string to lowercase.

Syntax:

```
strlwr (char *s1);
```

Ex:

```
char s1[15] = "HELLO";
strlwr(s1);
puts(s1);
```

O/P:

hello

4. strupr()

- It converts lowercase strings to uppercase strings.

Syntax: `strupr (char *string);`

Ex:
```
char s1[15] = "hello";
strupr(s1);
puts(s1);
```

O/P: HELLO

5. strcat():

This function appends the target string to the source string.

Syntax: `strcat (char *source, char * target);`

Ex:
```
char s1[30] = "I am";
char s2[30] = "an Indian";
strcat(s1, s2);
puts(s1);
```

O/P: I am an Indian

Some other string handling functions are.

- strrev() - reversing all characters of a string.
- strcpy()
- stricmp()
- strncmp()
- strnicmp()
- strdup()

8. What is structure? Explain how to access members of structure using C program?

- Structure is a collection of different data type variables grouped together under a single name
- It is mechanism for packing data of different types.

Syntax:

```
struct tagname
{
    datatype member1;
    datatype member2;
    ..
    datatype membern;
};
```

Eg:

```
struct book
{
    int pages;
    char author[30];
    float price;
};
```

- The individual members can be ordinary variables.
- The members names within a particular structure must be distinct from one another
- Structure variable declaration

Method 1:

```
struct tagename
{
    datatype member1;
    ..
    datatype membern;
} var1, var2 .. varn;
```

Method 2:

```
struct tagname var1, var2, varn;
```

- tagname is an identifier
- structure members can be acced using dot(.) operator

Eg: `struct_var_name . member;`

- Structure members can be accessed by * & → operator also

Program:

```
#include <stdio.h>
struct student
{
   int roll;
   char name[30];
   float marks;
};
int main()
{
   struct student s1;
   s1.roll = 101;
   strcpy(s1.name, "Ravi");
   s1.marks = 88.5;
   printf("Roll = %d \n", s1.roll);
   printf("name = %s \n", s1.name);
   printf("Marks = %.2f", s1.marks);
   return(0);
}
```

9a) What is a pointer? How pointer is declared and initialized.

Definition:

- Pointer is a variable it store address of another variable.
- A pointer "point to" a location in memory where some data is stored.
- Using pointers, a program can access and modify that data indirectly by using the stored address rather

than the variable's name.

→ Pointer Declaration

Syntax

```
data_type *pointer_name;
```

Eg: int *ptr;

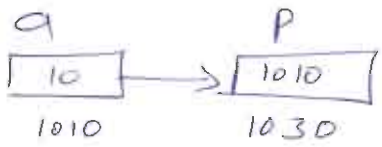
The data_type specifies what type of variable the pointer can point to. and '*' indicates that the variable is a pointer.

→ Pointer initialization with address

- A pointer is usually initialized with the address of a variable using the address-of operator &.

Eg:

```
int a = 10;
int *p;
p = &a;
```



- Declaration and Initialisation can be combined.

```
int a=10;
int *p = &a;
```

- other common initialization

```
int *p = NULL;
```

this clearly indicates that 'p' does not currently point to a valid address.

- one pointer can also be initialized from another pointer of the same type

```
int *p1 = &a;
int *p2 = p1;
```

- Any Example program.

9b) Differentiate between structure and union

Structure	Union
- Definition It is heterogeneous collection of data items grouped together under a single name	- Definition It is a memory location that is shared by several variables of different data types.

- Syntax:
struct tagname
{
datatype member1;
datatype member2;
};
- Keyword : struct
struct sample
	- Syntax
	union tagname
	{ datatype member1;
	datatype member2;
	};
	- Keyword : union

- All members of a structure can be initialized when structure variable is declared	- In union, we can initialize only one member when union variable is declared.
- Memory allocation: Memory allocated is the sum of sizes of all the data types in structure	- Memory allocation Memory allocated is the maximum size allocated among all the data types in union
- Memory is allocated for all the members of the structure differently	- Mem Only one member will be residing in the memory at any particular instance

10 What is sorting? Explain Bubble sort with an example?

- Bubble sort is the simplest sorting algorithm that works by ~~repeatd~~ repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets.
- It is called bubble sort because the movement of array elements is just like the movement of air bubbles in the water

Algorithm

Bubble_sort (a, n)

step 1: start

step 2: set i=0, repeat step 3 until i<n-1, increase 'i' value by 1

step 3: set j=0, repeat step 4 until j<n-1-i, increase 'j' value by 1

step 4: if a[j] > a[j+1]

set temp = a[j];

set a[j] = a[j+1];

set a[j+1] = temp;

[end of if]

[end of loop (step 3)]

[end of loop (step 2)]

step 5: end

Example:

Sort the following elements 50, 25 and 5 - by using

Pass-1

- Compare 50 and 25

Adjacent Pairs	Compare	swap	After swap
50 | 25 | 5	50 > 25	yes	25 | 50 | 5
25 | 50 | 5	50 > 5	yes	25 | 5 | 50 (sorted)

Pass-2

Adjacent Pairs	Compare	swap	After swap
25 | 5 | 50	25 > 5	yes	5 | 25 | 50 (sorted)

So the array is completely sorted.

11. Write an algorithm to implement linear search in an array with an example.

Sol:- - Linear search is also called as sequential search algorithm.

- It is a simple searching algorithm that checks each element in the list one by one until the desired item is found or end of the list is reached.

Algorithm

Step 1: Start

Step 2: Read the search element (Target element) in the array.

Step 3: Compare the search element with the first element in the array.

Step 4: If both are matched, terminate the process

Step 5: If both are not matched, Compare the search element with the next element in the array.

Step 6: Repeat step 5 until the search element is compared with the last element of the array.

Step 7: If the last element in the list does not match, print "element is not found" stop

Step 8: stop

Algorithm

Linear_search(a, n, key)

Step 1 : set pos = -1;

Step 2 : set i=0;

Step 3 : repeat step 4 until i<n, 'i' increment by 1

Step 4 : if a[i] == key

set pos = i;

print "pos"

goto step 6

[end of if]

set i = i+1;

[end of loop]

Step 5 : if pos = -1

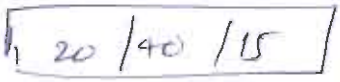
print " Value is not present in the list"

[end of if]

Step 6 : end.

Example.

Given array

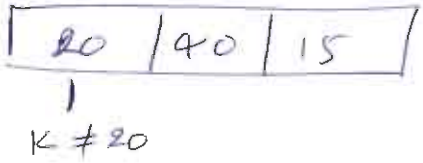


key = 40

Step 1:

. Compare key 40 with first element 20

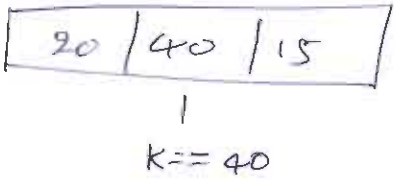
. not equal, move to next element.



Step 2:

Compare key 40 with second element

. Match found, so search is successful



∴ Search is successful.

— x — x —

CMR Engineering College

Subject: PPS Reg: R25

I B.Tech I Sem (Regular) Dec - 2025

SCHEME

PART-A

1a) Printf() — 1M
scanf() — 1M

b) Actural Parameter definition — 1M
Formal " " — 1M

c) gets() purpose — 1M
puts() " — 1M

d) Definition — 2M

e) Any 2 differences — 2M

PART-B

2a) Each section — 1M 1X5 - 5M

2b) Any Five differences — 1M X5 — 5M

3 if-statement — 1M
if-else statement — 3M
nested if " — 3M
switch " — 3M

4 Function Definition & Syntax — 5M
Program — 5M

5 Auto storage class — 2.5 M

Static " " — 2.5 M

extern " " — 2.5 M

register " " — 2.5 M

6 Definition & Syntax — 2M

Declaration of arrays — 2M

Logic — 5M

~~Eighth~~ Without errors — 1M

7 Definition — 2M

Any 5 string handling function — 8M

8 Definition & Syntax — 4M

Program — 6M

9a) Pointer definition ~~&~~ Syntax — ~~3~~ 2M

declaration & initialization — 3M

b) Any 5 differences — 5M

10) Sorting definition — 2M

Algorithm — 4M

Example — 4M

11) Algorithm — 5M

Example — 5M

— x — x —