B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

1a) Convert the decimal number 59 into its equivalent binary, octal, and hexadecimal equivalent.

Convert 59 to Binary (Base 2)

Binary uses only **0s and 1s**. We repeatedly divide by 2 and record the remainders.

Division by 2	**Quotient**	**Remainder**
59 ÷ 2 = 29	29	**1**
29 ÷ 2 = 14	14	**1**
14 ÷ 2 = 7	7	**0**
7 ÷ 2 = 3	3	**1**
3 ÷ 2 = 1	1	**1**
1 ÷ 2 = 0	0	**1**

Now, we **read from bottom to top**:
59_{10} = 111011_2

---

2. Convert 59 to Octal (Base 8)

Octal uses digits **0-7**. We repeatedly divide by 8 and record the remainders.

Division by 8	Quotient	Remainder
59 ÷ 8 = 7	7	**3**
7 ÷ 8 = 0	0	**7**

Now, we **read from bottom to top**:
59_{10} = 73_8

---

3. Convert 59 to Hexadecimal (Base 16)

Hexadecimal uses digits **0-9 and A-F**. We repeatedly divide by 16 and record the remainders.

Division by 16	Quotient	Remainder
59 ÷ 16 = 3	3	**11 (B)**

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

3 ÷ 16 = 0 0 3

Now, we read from bottom to top:
59_{10} = 3B$_{16}$ (where B represents 11 in decimal)

Final Answer:

Binary: 111011_2
Octal: 73_8
Hexadecimal: $3B_{16}$

1b) Explain the difference between implicit and explicit type conversion in C.

Feature	**Implicit Type Conversion (Type Promotion)**	**Explicit Type Conversion (Type Casting)**
Definition	Automatically performed by the compiler.	Manually done by the programmer using type casting.
Syntax	Happens implicitly, no special syntax needed.	Uses (type) before the variable or expression (e.g., (float)
Example	float result = 5 + 2.5; (5 is converted to float)	float result = (float)5 / 2; (5 is explicitly cast to float)
Risk of Data Loss	No, as smaller types are promoted safely.	Yes, when converting larger types to smaller ones (e.g., double to int).

1c) Compare while and do while statement with example

	While	do while
Definition	A loop that executes **while** a condition is true.	A loop that executes **at least once**, then continues if the condition is true.
Condition Check	Condition is checked **before** execution.	Condition is checked **after** execution.
Execution Guarantee	May not execute if the condition is false at the beginning.	Always executes at least **once**, even if the condition is false
Syntax	c while(condition) { // Code }	c do { // Code } while(condition);

Example	int x = 5;
	while (x > 10)
	{
	printf("Hello");
	x++;
	} // Does NOT execute
		int x = 5; do
		{
		printf("Hello");
		x++;
		}
		while (x > 10); // Executes

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

1 d) Write a C program to calculate factorial of a number using recursion

// C program to find factorial of given number using recursion//

```
#include <stdio.h>
unsigned int factorial(unsigned int n)
{
   // Base Case//
   if (n == 1)
 {
      return 1;
  else
   // Multiplying the current N with the previous product of Ns//
   return n * factorial(n - 1);
}

int main() {
   int num = 5;
   printf("Factorial of %d is %d", num, factorial(num));
   return 0;
}
```

Output:

Factorial of 5 is 120

1e) Compare structure and union based on memory usage and use cases in c programming

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

Parameter	Structure	Union
Definition	A structure is a user-defined data type that groups different data types into a single entity.	A union is a user-defined data type that allows storing different data types at the same memory location.
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union
Size	The size is the sum of the sizes of all members, with padding if necessary.	The size is equal to the size of the largest member, with possible padding.
Memory Allocation	Each member within a structure is allocated unique storage area of location.	Memory allocated is shared by individual members of union.
Data Overlap	No data overlap as members are independent.	Full data overlap as members shares the same memory.
Accessing Members	Individual member can be accessed at a time.	Only one member can be accessed at a time.

1f) Define pointer variable, give an example of NULL pointer

Pointer variable: A **pointer** is a variable that **stores** the **memory address** of another variable as its value.

NULL pointer:

A null pointer in C does not point to any memory location.

- They are used in dynamic memory allocation, error handling, etc. Any pointer which is assigned the value NULL becomes a null pointer.

//C NULL pointer demonstration//

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

```
#include <stdio.h>

int main()

{

   // declaring null pointer

   int* ptr = NULL;

   // derefencing only if the pointer have any value

   if (ptr == NULL) {

      printf("Pointer does not point to anything");

   }

   else {

      printf("Value pointed by pointer: %d", *ptr);

   }

   return 0;

}
```

Output

Pointer does not point to anything

2a) Define operator? How many types of operators, Explain any four types of operators with examples.

OPERATORS

C supports a rich set of operators. Operators are used in programs to manipulate data and variables. They usually form a part of the mathematical of logical expressions.

C operators are classified into a number of categories.
They include:

1. Arithmeticoperators
2. Relationaloperators

3. Logicaloperators
4. Assignmentoperators
5. Increment and Decrementoperators
6. Conditionaloperators
7. Bitwiseoperators
8. Specialoperators

ARITHMETIC OPERATORS

The Arithmetic operators are

+ (Addition)
- (Subtraction)
* (Multiplication)
/ (Division)
% (Modulo division)

Eg: 1) a-b 2) a+b3) a*b 4)a/b 5)a%b

The modulo division produces the remainder of an integer division. The modulo division operator cannot be used on floating pointdata.

Note: C does not have any operator for exponentiation.

RELATIONAL OPERATORS

Comparisons can be done with the help of relational operators. The expression containing a relational operator is termed as a relational expression.The value of a Relational Expression is either zero or 1.

Operater	meaning
1) <	(is less than)
2) <=	(is less than or equal to)
3) >	(is greater than)

4) >= (is greater than or equal to)

5) = = (is equal to)

6) != (is not equal to)

Eg: a<b or1<20

LOGICAL OPERATERS

C has the following three logical operators.

&& (logicalAND)

|| (logical OR)

! (logicalNOT)

Eg: 1) if(age>55 &&sal<1000)

2) if(number<0 || number>100)

ASSIGNMENT OPERATORS

The usual assignment operator is '='.In addition, C has a set of 'shorthand' assignment operators of the form, v op = exp;

Eg: x +=y+1;

This is same as the statement

x=x+(y+1);

2b) Explain the significance of each number system (decimal, binary, octal, and hexadecimal) in computer applications.Convert 478 into binary, hexadecimal and Convert 2F3A into binary and octal number

Convert 478 into binary, hexadecimal

---

1. Convert 478 to Binary (Base 2)

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

Binary uses only **0s and 1s**. We repeatedly **divide by 2** and record the remainders.

Division by 2	Quotient	Remainder
478 ÷ 2 = 239	239	**0**
239 ÷ 2 = 119	119	**1**
119 ÷ 2 = 59	59	**1**
59 ÷ 2 = 29	29	**1**
29 ÷ 2 = 14	14	**1**
14 ÷ 2 = 7	7	**0**
7 ÷ 2 = 3	3	**1**
3 ÷ 2 = 1	1	**1**
1 ÷ 2 = 0	0	**1**

Now, we **read from bottom to top**:
478_{10} = 111011110_2

---

2. Convert 478 to Hexadecimal (Base 16)

Hexadecimal uses digits **0-9 and A-F**. We repeatedly **divide by 16** and record the remainders.

Division by 16	Quotient	Remainder
478 ÷ 16 = 29	29	**14 (E)**
29 ÷ 16 = 1	1	**13 (D)**
1 ÷ 16 = 0	0	**1**

Now, we **read from bottom to top**:
478_{10} = $1DE_{16}$ (where **D = 13** and **E = 14** in decimal)

Final Answer:

- **Binary:** 111011110_2
- **Hexadecimal:** $1DE_{16}$

Convert 2F3A into binary and octal number

Let's convert **2F3A (hexadecimal)** into **binary and octal**, step by step.

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

1. Convert 2F3A (Hex) to Binary (Base 2)

Each **hexadecimal digit** is converted into a **4-bit binary equivalent**:

Hex Digit	**Binary Equivalent**
2	0010
F (15)	1111
3	0011
A (10)	1010

Now, combining all the binary values:
$2F3A_{16}$ = 0010 1111 0011 1010$_2$
Removing leading zeros: **10111100111010_2**

---

2. Convert 2F3A (Hex) to Octal (Base 8)

To convert hex to octal, we first convert **hex to binary**, then group the binary digits into sets of **three** (starting from the right).

We already have:
$2F3A_{16}$ = 10111100111010_2

Now, grouping in sets of **three** from right to left:

10 111 100 111 010 (add leading 0s if needed: 010 111 100 111 010)

Now, convert each **3-bit group** to octal:

Binary Group	**Octal Equivalent**
010	2
111	7
100	4
111	7
010	2

So,
$2F3A_{16}$ = 27472_8

Final Answer:

- **Binary:** 10111100111010_2
- **Octal:** 27472_8

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

3a) Define array. Write a C program to perform matrix multiplication of two 2 dimensional arrays using nested loop. Ensure that matrix dimensions are validated before performing multiplication.

An array is a group of related data items that share a common name and common type.

eg: An array name is ‘salary’s used to represent a set of salaries of a group of employees.

- A particular value is indicated by writing a number ‘índex’or ‘subscript’ in brackets after the array name.

Eg:salary[10]

Advantage: The ability to use a single name to represent a collection of items and to refer to an item by specifying the item number enables to develop concise and efficient programs.

PROGRAM

```
#include <stdio.h>
int main() {
  int r, c, a[100][100], b[100][100], sum, i,k,mul[10][10], j;
printf("Enter the number of rows (between 1 and 100): ");
scanf("%d", &r);
printf("Enter the number of columns (between 1 and 100): ");
scanf("%d", &c);
printf("\nEnter elements of 1st matrix:\n");
  for (i = 0; i< r; i++)
    for (j = 0; j < c; j++) {
printf("Enter element a%d%d: ", i + 1, j + 1);
scanf("%d", &a[i][j]);
    }
printf("Enter elements of 2nd matrix:\n");
  for (i = 0; i< r; i++)
    for (j = 0; j < c; j++) {
printf("Enter element b%d%d: ", i + 1, j + 1);
scanf("%d", &b[i][j]);
    }
```

```
 // adding two matrices and printing sum of matrices
 for (i = 0; i< r; i++)
   for (j = 0; j < c; j++) {
            for(k=0;k<r;k++)
            {
            sum+=  a[i][k] * b[k][j];
}
mul[i][j]=sum;
 sum=0;
printf("%d   ", mul[i][j]);
printf("\n\n");
     }
   }
```

3b) write a C program to reverse the given integer number

```
#include <stdio.h>

int main() {
  int n, reverse = 0, remainder, original;

printf("Enter an integer: ");
scanf("%d", &n);

  original = n;

  while (n != 0) {
    remainder = n % 10;
    reverse = reverse * 10 + remainder;
    n /= 10;
  }
```

```
  if (original % 10 == 0) {
printf("Reversed number = %d", reverse);


    while (original % 10 == 0) {
printf("0");
      original /= 10;
    }
  } else {
printf("Reversed number = %d", reverse);
  }


  return 0;
}
```

Output:

Enter an integer: 2345

Reversed number = 5432

4a) Define function? Explain function call, function definition and function prototype with example

Functions

A function is a block of code that performs a specific task.

Dividing complex problem into small components makes program easy to understand and use.

Types of functions

Depending on whether a function is defined by the user or already included in C compilers, there are two types of functions in C programming

Standard library functions

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

User defined functions

Elements of user defined functions

1.function definition

2. function call

3. function declaration or prototype

Function definition

Function definition contains the block of code to perform a specific task i.e. in this case, adding two numbers and returning it.

It is divided into two parts namely **function head, function body**.

When a function is called, the control of the program is transferred to the function definition. And, the compiler starts executing the codes inside the body of a function.

Note: We should not use semicolon **;** in function definition() head same like main() function.

The general syntax of a functiondefinition is as follows:

```
return_typefunction_name(parameter list)  /*……..function  head…….*/
{
        local_variables declaration;    /*…..function body…..*/
        executable  statement(s);
return_value;
    }
```

Example:

```
int add( int a, int b )
{
        int sum;
        sum = a +  b;
```

return sum; }

Syntax of return statement

```
return (expression);
```

For example,

```
return a;

return (a+b);
```

The type of value returned from the function and the return type specified in function prototype and function definition must match.

Function Call

Control of the program is transferred to the user-defined function by calling it.

Syntax of function call

```
functionName(argument1, argument2, ...);
```

In the above example, function call is made using addNumbers(n1,n2); statement inside the main().

Function Declaration :

- **Like variables function must be declared before they are used.**
- **A function prototype gives information to the compiler that the function may later be used in the program.**

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

It consists of 4 parts.

- **Function type**
- **Function name**
- **Parameter list**
- **Terminating semi colon**

SYNTAX:

returntypefunctionname(type1 argument1, type2 argument2,...);

Example:

int mul(int a,int b);

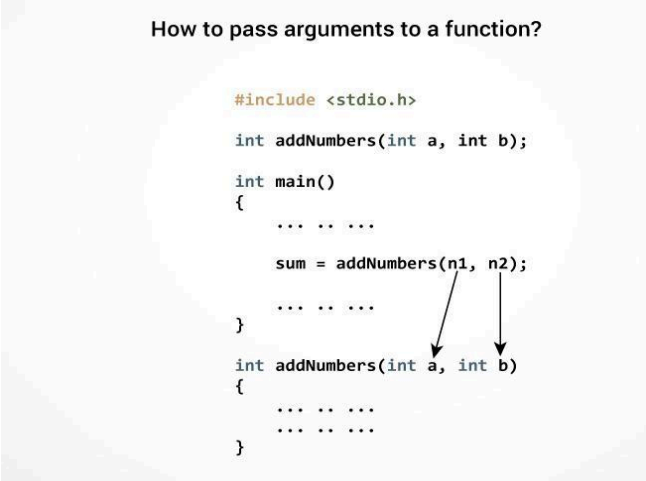
int mul(int,int);

Passing arguments to a function

In programming, argument refers to the variable passed to the function. In the above example, two variables n1 and n2 are passed during function call. These are called actual parameters.

The parameters a and b accepts the passed arguments in the function definition. These arguments are called formal parameters of the function.

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)



The type of arguments passed to a function and the formal parameters must match, otherwise the compiler throws error.

If `n1` is of char type, `a` also should be of char type. If `n2` is of float type, variable `b` also should be of float type.

A function can also be called without passing an argument.

4b) write a C program to check whether the sum of the digits of a number is even or odd.

```
#include <stdio.h>

int main() {
    int num, sum = 0, digit;

    // Input a number from the user
```

```
printf("Enter a number: ");
scanf("%d", &num);

    // Calculate the sum of digits
    while (num> 0) {
        digit = num % 10;  // Extract the last digit
        sum += digit;      // Add it to the sum
num /= 10;         // Remove the last digit
    }

    // Check if the sum is even or odd
    if (sum % 2 == 0) {
printf("The sum of the digits is EVEN.\n");
    } else {
printf("The sum of the digits is ODD.\n");
    }

    return 0;
}
```

---

5 a) Define structure and union. Explain how member of structure and union accesses using program code

Structure Definition:

Structure is a user defined data type which hold or store heterogeneous/different types data item or element in a single variable. It is a Combination of primitive and derived data type.

or

A structure is a collection of one or more data items of different data types, grouped together under a single name. Variables inside the structure are called members of structure.

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

Union Defination:

A union is a special data type available in C that allows to store different data types in the same memory location.

The members of a structure are accessed outside the structure by the structure variables using the dot operator (.). The following syntax is used to access any member of a structure by its variable:

Syntax

structVariable.structMember

Example

The following example illustrates how to access the members of a structure and modify them in C.

```
#include <stdio.h>

#include <string.h>

struct cube

{

  // data members

  char P_name[10];

  int P_age;

  char P_gender;

};

int main()

{

  // structure variables

  struct cube p1, p2;
```

```
// structure variables accessing the data members.

strcpy(p1.P_name, "XYZ");

p1.P_age = 25;

p1.P_gender = 'M';

strcpy(p2.P_name, "ABC");

p2.P_age = 50;

p2.P_gender = 'F';

// print the patient records.

// patient 1

printf("The name of the 1st patient is: %s\n", p1.P_name);

printf("The age of the 1st patient is: %d\n", p1.P_age);

if (p1.P_gender == 'M')

{

  printf("The gender of the 1st patient is: Male\n");

}

else

{

  printf("The gender of the 1st patient is: Female\n");

}

printf("\n");

// patient 2
```

```
  printf("The name of the 2nd patient is: %s\n", p2.P_name);

  printf("The age of the 2nd patient is: %d\n", p2.P_age);

  if (p2.P_gender == 'M')

  {

    printf("The gender of the 2nd patient is: Male\n");

  }

  else

  {

    printf("The gender of the 2nd patient is: Female\n");

  }

  return 0;

}
```

Access members of a union and structure:

We use the . operator to access members of a union and structure.
to access pointer variables, we use the -> operator.

Example: Accessing Union Members

```
#include <stdio.h>

union Job {

  float salary;

  int workerNo;

} j;


int main() {

  j.salary = 12.3;
```

```
  // when j.workerNo is assigned a value,
  // j.salary will no longer hold 12.3
  j.workerNo = 100;

  printf("Salary = %.1f\n", j.salary);
  printf("Number of workers = %d", j.workerNo);
  return 0;
}
```

Output

Salary = 0.0

Number of workers = 100

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

struct Person {
    char name[50];
    int age;
    float height;
};

int main() {
    struct Person person1;
```

```
    strcpy(person1.name, "John Doe");
    person1.age = 25;
    person1.height = 6.1;

    printf("Name: %s\n", person1.name);
    printf("Age: %d\n", person1.age);
    printf("Height: %.2f\n", person1.height);

    return 0;
}
```

Output:

Name: John Doe

Age: 25

Height: 6.10

5b)write a c program to maintain a record of n students details using an array of structures with four fields(roll number,name,marks and grade).Assume appropriate data type for each field. Print the marks of the student give the student name as input

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>
struct student{
 char name[14];
 int rollNo;
 int marks[3];
 float percentage;
};

int main(){
 struct student *stuArray;
 int n,i,j;
 int tempTotal=0,flag=0,foundIndex;
 char tempName[14];
 clrscr();
```

```
printf("\n No of Students: ");
scanf("%d",&n);
stuArray = (struct student*)malloc(n*sizeof(struct student));
for(i=0;i<n;i++){
 printf("\n %d.Name :",(i+1));
 scanf("%s",&stuArray[i].name);
 printf("\n Roll Number :",(i+1));
 scanf("%d",&stuArray[i].rollNo);
 tempTotal=0;
 for(j=0;j<3;j++)
 {
  printf("\n Mark %d :",(j+1));
  scanf("%d",&stuArray[i].marks[j]);
  tempTotal+=stuArray[i].marks[j];
 }
 stuArray[i].percentage=tempTotal/3;
}

printf("\n Enter the Name to be Searched: ");
scanf("%s",&tempName);
for(i=0;i<n;i++){
 if(strcmp(tempName,stuArray[i].name)==0)
 {
  foundIndex=i;
  flag=1;
  break;
 }
}
if(flag==0)
{
 printf("Details Not Found");
}
else
{
 printf("\n Mark of %s are given below...",tempName);
 for(i=0;i<3;i++)
 {
  printf("\n Mark %d is %d",(i+1),stuArray[foundIndex].marks[i]);
 }
}
getch();
return 0;
}
```

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

6a) Write a c program to find the sum and mean of all elements in an array using pointer

```
#include <stdio.h>
int main()
{
	int Size, i, sum = 0;

	printf("Please Enter the Array size = ");
	scanf("%d", &Size);
	int arr[Size];
	int *parr = arr;
	printf("Enter the Array Items = ");
	for (i = 0; i < Size; i++)
	{
		scanf("%d", parr + i);
	}
	for (i = 0; i < Size; i++)
	{
		sum = sum + *(parr + i);
	}
	float avg = (float)sum / Size;
	printf("\nThe Sum of Array Items     = %d\n", sum);
	printf("\nThe Average of Array Items = %.2f\n", avg);
}
```

6b) Develop a c program that reads an employee data to store into a file and allow user to search for an employee by id, displaying their if found

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

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

```
#include <stdlib.h>
#include <string.h>
#define FILENAME "employees.dat"
// Employee structure
typedef struct {
    int id;
    char name[50];
    float salary;
} Employee;

// Function to add employee data to file
void addEmployee() {
    Employee emp;
    FILE *file = fopen(FILENAME, "ab"); // Append mode

    if (file == NULL) {
        printf("Error opening file!\n");
        return;
    }

    printf("Enter Employee ID: ");
    scanf("%d", &emp.id);
    printf("Enter Employee Name: ");
    scanf("%s", emp.name);
    printf("Enter Employee Salary: ");
    scanf("%f", &emp.salary);

    fwrite(&emp, sizeof(Employee), 1, file);
    fclose(file);
    printf("Employee added successfully!\n");
```

```
}

// Function to search for an employee by ID
void searchEmployee() {
    int id;
    Employee emp;
    FILE *file = fopen(FILENAME, "rb"); // Read mode

    if (file == NULL) {
        printf("Error opening file!\n");
        return;
    }

    printf("Enter Employee ID to search: ");
    scanf("%d", &id);

    while (fread(&emp, sizeof(Employee), 1, file)) {
        if (emp.id == id) {
            printf("Employee Found:\n");
            printf("ID: %d\n", emp.id);
            printf("Name: %s\n", emp.name);
            printf("Salary: %.2f\n", emp.salary);
            fclose(file);
            return;
        }
    }

    printf("Employee with ID %d not found!\n", id);
    fclose(file);
}
```

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

```
int main() {
    int choice;
    do {
        printf("\nEmployee Management System\n");
        printf("1. Add Employee\n");
        printf("2. Search Employee by ID\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addEmployee();
                break;
            case 2:
                searchEmployee();
                break;
            case 3:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice! Please enter again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

7a) What is dynamic array? How it is created? Give a typical example of use of a dynamic array

- A Dynamic Array is allocated memory at runtime and its size can be changed later in the program.

We can create a dynamic array in C by using the following methods:

Using malloc() Function

It is defined inside **<stdlib.h>** header file.

Syntax:

```
ptr = (cast-type*) malloc(byte-size);
```

Using calloc() Function

Syntax:

```
ptr = (cast-type*)calloc(n, element-size);
```

Resizing Array Using realloc() Function

Syntax:

```
ptr = realloc(ptr, newSize);
```

```
// C program to create dynamic array using malloc() function

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

int main()
{

    // address of the block created hold by this pointer
    int* ptr;
    int size;

    // Size of the array
    printf("Enter size of elements:");
    scanf("%d", &size);

    //  Memory allocates dynamically using malloc()
    ptr = (int*)malloc(size * sizeof(int));

    // Checking for memory allocation
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
```

```
    }
    else {

        // Memory allocated
        printf("Memory successfully allocated using "
               "malloc.\n");

        // Get the elements of the array
        for (int j = 0; j < size; ++j) {
            ptr[j] = j + 1;
        }

        // Print the elements of the array
        printf("The elements of the array are: ");
        for (int k = 0; k < size; ++k) {
            printf("%d, ", ptr[k]);
        }
    }

    return 0;
}
```

Output:

```
Enter size of elements:5
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5,
```

7b) Write a c program to find the transpose of a matrix. Explain the logic and provide an example with input and output

```
#include <stdio.h>

int main() {

  int a[10][10], transpose[10][10], r, c;

  printf("Enter rows and columns: ");

  scanf("%d %d", &r, &c);


  // asssigning elements to the matrix

  printf("\nEnter matrix elements:\n");

  for (int i = 0; i < r; ++i)

  for (int j = 0; j < c; ++j) {

    printf("Enter element a%d%d: ", i + 1, j + 1);
```

```
  scanf("%d", &a[i][j]);
 }

 // printing the matrix a[][]
 printf("\nEntered matrix: \n");
 for (int i = 0; i < r; ++i)
 for (int j = 0; j < c; ++j) {
  printf("%d  ", a[i][j]);
  if (j == c - 1)
  printf("\n");
 }

 // computing the transpose
 for (int i = 0; i < r; ++i)
 for (int j = 0; j < c; ++j) {
  transpose[j][i] = a[i][j];
 }

 // printing the transpose
 printf("\nTranspose of the matrix:\n");
 for (int i = 0; i < c; ++i)
 for (int j = 0; j < r; ++j) {
  printf("%d  ", transpose[i][j]);
  if (j == r - 1)
  printf("\n");
 }
 return 0;
}
```

Output

B.E I-SEMESTER REGULAR (MAIN)EXAMINATION JAN/FEB-2025
C PROGRAMMING
(Common to CIVIL & CSE)

Enter rows and columns: 2

3

Enter matrix elements:

Enter element a11: 1

Enter element a12: 4

Enter element a13: 0

Enter element a21: -5

Enter element a22: 2

Enter element a23: 7

Entered matrix:

1 4 0

-5 2 7

Transpose of the matrix:

1 -5

4 2

0 7