Lab exercise:-

(1) - (1) - (i) Embedded Systems

- Example: Washing machines, microwave ovens, car control systems (like ABS).
- Why C is used:
  - C provides direct access to hardware via pointers.
  - It has low memory usage and fast execution.
- Real-world Use Case:
  - Automobiles use C to program Electronic Control Units (ECUs) that manage braking, lighting, fuel injection, etc.

(ii) Operating Systems

- Example: Windows, Linux, UNIX.
- Why C is used:
  - C provides low-level access to memory and system processes.
  - It's highly portable and efficient.
- Real-world Use Case:
  - The Linux Kernel is written in C, which manages process scheduling, memory, and device control.

(iii) Game Development

- Example: Doom (1993), Quake, and many early console games.
- Why C is used:
  - Offers high performance and fine control over memory.
  - Useful in game engines and graphics processing.
- Real-world Use Case:
  - Game engines like id Tech (Quake) were built using C for performance-critical parts.

(2) -

```
#include<stdio.h>
#include<conio.h>
void main()
{
   cout<<"hello world"<<endl;

   return 0;
}
```

(3) -

```
#include <stdio.h> // header file
int main() { // main function
// this is a comment
int a = 10; // variable declaration
printf("Value of a is %d", a);
return 0;
}
```

Explanation:

#include <stdio.h>: Header file
main(): Starting point of the program
//: Comment line
int: Data type
a: Variable nam

(4) -

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,b,result;
    //arthimetic operations..
    cout<<"enter a:"<<endl;
    cin>>a;
    cout<<"enter b:"<<endl;
    cin>>b;
    result=a+b;
    result=a*b;
    result=a/b;
    result=a-b;

    return 0;


}
```

(5) - if:

```
if (a > 0) {
printf("Positive number");
}
if-else:
if (a > 0) {
printf("Positive");
} else {
printf("Non-positive");
}
nested if-else:
if (a > 0) {
if (a < 100) {
printf("Between 1 and 99");
}
}
switch:
int choice;
switch (choice)
{
    case 1: january;
     break;
    case 2: february;
     break;
    case 3: march;
     break;
    case 4: april;
     break;
    case 5: may;
     break;
    case 6: june;
     break;
    case 7: july;
     break;
    case 8: august;
     break;
    case 9: september;
     break;
    case 10: october;
     break;
    case 11: november;
     break;
    case 12: december;
     break;
    default:
    invalid choice;
}
```

(6) -

while loop:

```
int i = 0;
while (i < 5)
 {
printf("%d\n", j);
i++;
}
```

Use when the number of iterations is unknown.

for loop:

```
for (int i = 0; i < 5; i++)
{
printf("%d\n", j);
}
```

Use when the number of iterations is known.

do-while loop:

```
int i = 0;
do {
printf("%d\n", i);
i++;
} while (i < 5);
```

(7) -.

break: Stops the loop.

```
for (int i = 0; i < 10; i++)
{
if (i == 5) break;
printf("%d\n", i);
}
```

continue: Skips the current iteration.

```
for (int i = 0; i < 5; i++)
{
if (i == 2) continue;
printf("%d\n", i);
}
```

(8) -

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

    // Function Declaration
    long long int calculateFactorial(int n);

    int main()
   {
        int number;
        long long int factorialResult;

    // Prompt user for input
    printf("Enter a non-negative integer: ");
    scanf("%d", &number);

   // Input validation
    if (number < 0)
    {
    printf("Factorial is not defined for negative numbers.\n");
    }
    else
    {
    // Function Call
    factorialResult = calculateFactorial(number);
    printf("The factorial of %d is %lld.\n", number, factorialResult);
}

    return 0;
}

 // Function Definition
 long long int calculateFactorial(int n)
 {
    long long int fact = 1;
    int i;

// Calculate factorial iteratively
    for (i = 1; i <= n; i++)
   {
        fact *= i;
    }
    return fact;
 }
```

(9) -

```
#include <stdio.h>
int main()
{
    // One-dimensional array
    int oneDArray[5] = {10, 20, 30, 40, 50};
    printf("Elements of one-dimensional array:\\n");
    for (int i = 0; i < 5; i++)
    {
        printf("%d ", oneDArray[i]);
    }
    printf("\\n\\n");

        int twoDArray[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int sum = 0;

    printf("Elements of two-dimensional array (3x3 matrix):\\n");
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            printf("%d ", twoDArray[i][j]);
            sum += twoDArray[i][j];
        }
        printf("\\n");
    }
    printf("\\nSum of all elements in the 3x3 matrix: %d\\n", sum);

 return 0;
}
```

(10) -

```
#include <stdio.h>
int main()
{
    int myVariable = 10; // Declare and initialize an integer variable
    int *ptr; // Declare an integer pointer
    printf("Initial value of myVariable: %d\n", myVariable);
    ptr = &myVariable; // Assign the address of myVariable to the pointer ptr
    // Modify the value of myVariable using the pointer
    *ptr = 25;
    printf("Value of myVariable after modification using pointer: %d\n", myVariable);
    return 0;
}
```

(11) -

```
#include <stdio.h>
#include <string.h> // Required for strcat() and strlen()

int main() {
char str1[100]; // Declare a character array to store the first string
char str2[50]; // Declare a character array to store the second string

printf("Enter the first string: ");
// Using fgets for safer input, handles spaces and prevents buffer overflow
fgets(str1, sizeof(str1), stdin);
// Remove the newline character potentially added by fgets
str1[strcspn(str1, "\n")] = '\0';

printf("Enter the second string: ");
fgets(str2, sizeof(str2), stdin);
// Remove the newline character potentially added by fgets
str2[strcspn(str2, "\n")] = '\0';

// Concatenate str2 to str1. Ensure str1 has enough allocated space.
strcat(str1, str2);

printf("\nConcatenated string: %s\n", str1);
printf("Length of the concatenated string: %zu\n", strlen(str1));

return 0;
}
```

(12) -

```
#include <stdio.h>

// Define the structure for a student
struct Student {
char name[50];
int rollNumber;
float marks;
};
int main() {
// Declare an array of 3 Student structures
struct Student students[3];

// Input student details
for (int i = 0; i < 3; i++) {
printf("Enter details for student %d:\n", i + 1);
```

```
    printf("Name: ");
    scanf("%s", students[i].name);
    printf("Roll Number: ");
    scanf("%d", &students[i].rollNumber);
    printf("Marks: ");
    scanf("%f", &students[i].marks);
 }
    // Print the student details
    printf("\n--- Student Details ---\n");
    for (int i = 0; i < 3; i++) {
    printf("Student %d:\n", i + 1);
    printf("Name: %s\n", students[i].name);
    printf("Roll Number: %d\n", students[i].rollNumber);
    printf("Marks: %.2f\n", students[i].marks);
    printf("----------------------\n");
}
    return 0;
 }
```

(13) -

```
    #include <stdio.h>
    #include <stdlib.h> // Required for exit()

    int main()
{
    FILE *fptr;
    char dataToWrite[] = "This is a test string written to the file.";
    char buffer[100]; // Buffer to store read data
    // 1. Create and Write to the file
    fptr = fopen("example.txt", "w"); // Open in write mode ("w")
    if (fptr == NULL) {
    printf("Error opening file for writing!\n");
    exit(1); // Exit if file cannot be opened
}
    fprintf(fptr, "%s", dataToWrite); // Write the string to the file
   fclose(fptr); // Close the file
    printf("String successfully written to example.txt\n");

    // 2. Open the file again and Read its contents
    fptr = fopen("example.txt", "r"); // Open in read mode ("r")
    if (fptr == NULL) {
    printf("Error opening file for reading!\n");
    exit(1); // Exit if file cannot be opened
    }
```

```
    printf("\nContents of example.txt:\n");
    // Read and print the contents line by line
    while (fgets(buffer, sizeof(buffer), fptr) != NULL) {
    printf("%s", buffer);
    }

    fclose(fptr); // Close the file after reading

    return 0;
}
   #include <stdio.h>
    #include <stdlib.h> // Required for exit()

    int main() {
    FILE *fptr;
    char dataToWrite[] = "This is a test string written to the file.";
    char buffer[100]; // Buffer to store read data

    // 1. Create and Write to the file
    fptr = fopen("example.txt", "w"); // Open in write mode ("w")
    if (fptr == NULL) {
    printf("Error opening file for writing!\n");
    exit(1); // Exit if file cannot be opened
    }
   fprintf(fptr, "%s", dataToWrite); // Write the string to the file
   fclose(fptr); // Close the file

    printf("String successfully written to example.txt\n");

    // 2. Open the file again and Read its contents
    fptr = fopen("example.txt", "r"); // Open in read mode ("r")
    if (fptr == NULL) {
    printf("Error opening file for reading!\n");
    exit(1); // Exit if file cannot be opened
    }

    printf("\nContents of example.txt:\n");
    // Read and print the contents line by line
    while (fgets(buffer, sizeof(buffer), fptr) != NULL) {
    printf("%s", buffer);
    }

    fclose(fptr); // Close the file after reading

    return 0;
    }
```