



C Programming Lab Report

(CACS-151)

Name: Mahendra Singh Mahara

Roll No: 15

Date of Submitted

.....

Signature

.....

Introduction to C Programming

C is a powerful, general-purpose programming language that has been widely used for decades. Developed in the early 1970s by Dennis Ritchie at Bell Labs, C is known for its efficiency, speed, and flexibility, making it the foundation for many modern programming languages like C++, Java, and Python.

Why Learn C?

- **Speed and Performance:** C allows low-level memory access, giving you fine control over system resources.
- **Portability:** C programs are highly portable and can run on different machines with minimal changes.
- **Foundation Language:** Learning C builds a strong base for understanding other languages and computer science concepts.
- **Widely Used in Systems Programming:** Operating systems, embedded systems, and performance-critical applications are often written in C.

Key Features of C

- Procedural language
- Rich set of built-in operators and functions
- Structured programming approach
- Dynamic memory management
- Modularity through functions

A Simple C Program

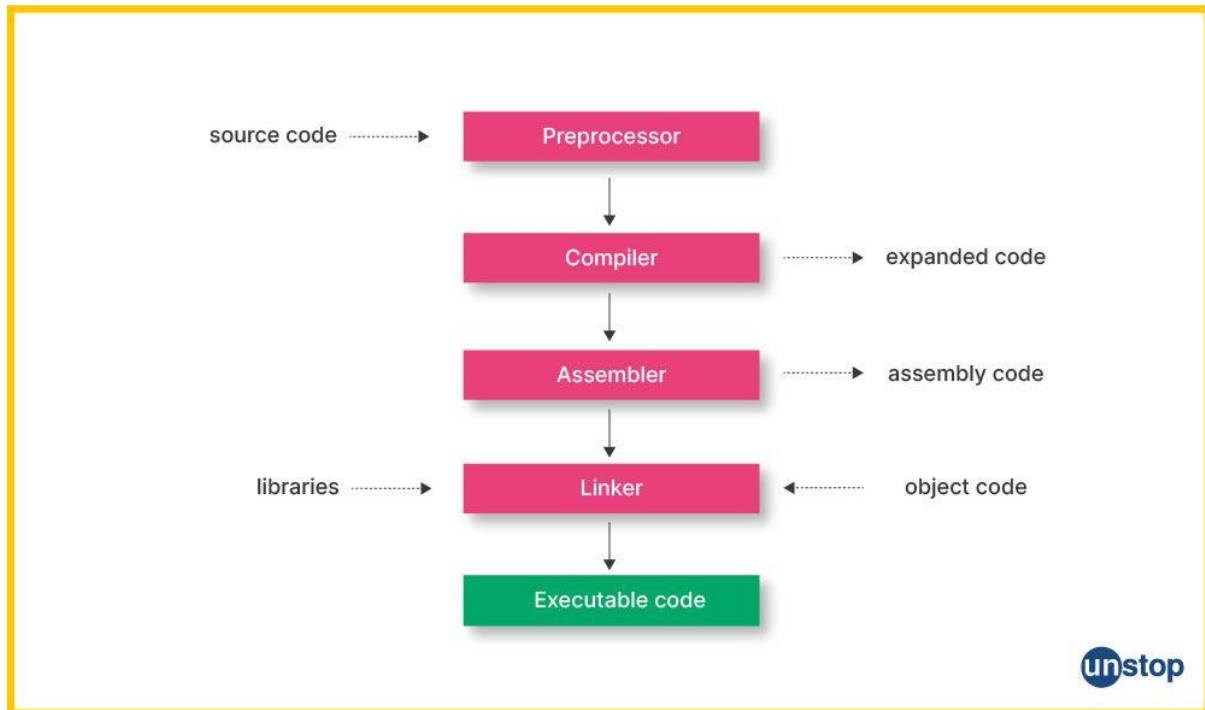
```
#include <stdio.h>
```

```
int main() {  
    printf("Hello, world!\n");  
    return 0;  
}
```

}

This program prints "Hello, world!" to the screen. It demonstrates the basic structure of a C program: inclusion of header files, the main() function, and use of standard I/O.

Compilation In C



Process of compiling and running a C program

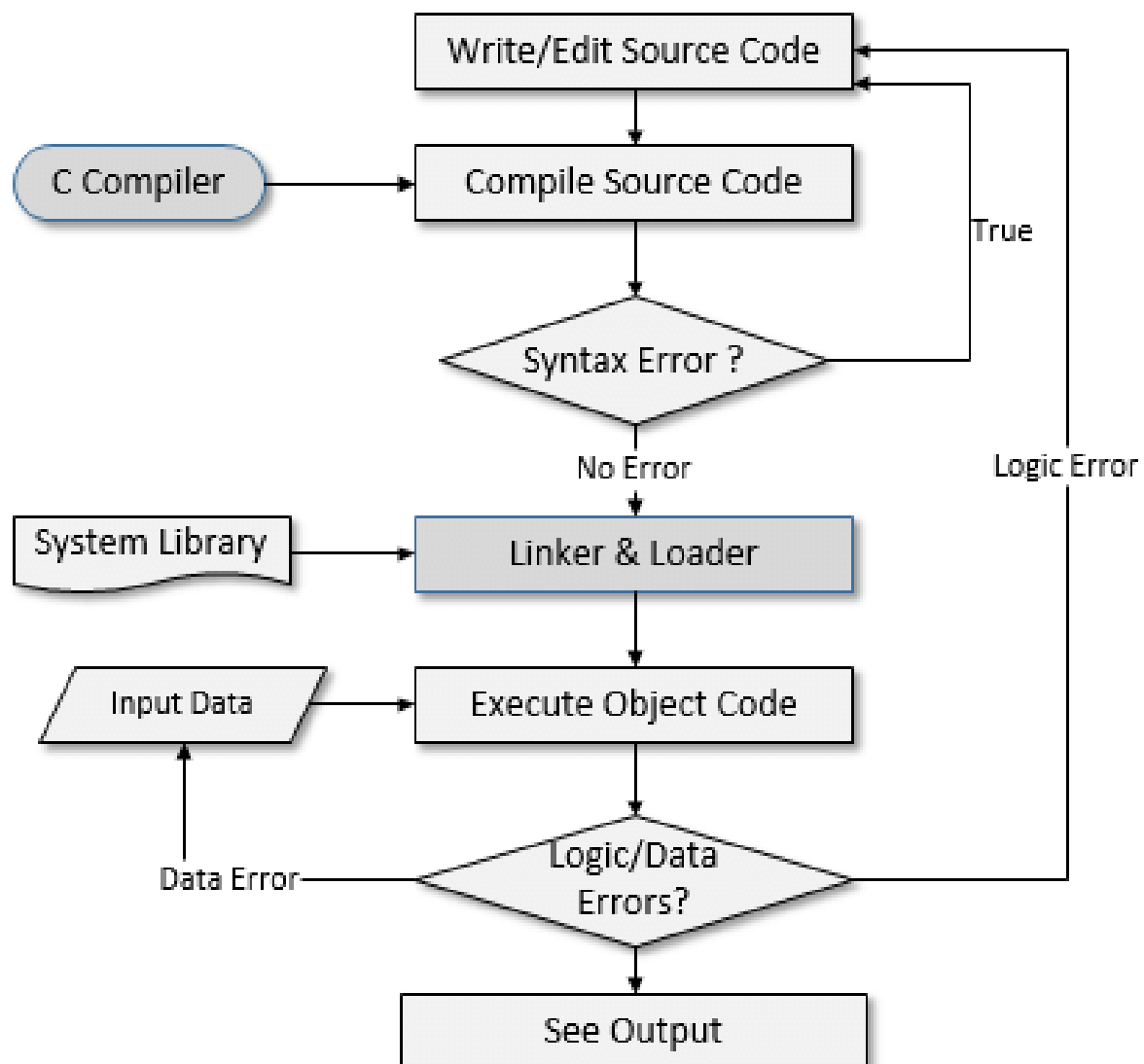
To run a C program, it must first go through several stages to convert the human-readable source code into machine-executable code. The process begins when a programmer writes the code using a text editor and saves it with a .c extension. This source code is then passed to the **preprocessor**, which handles all the lines beginning with #, such as #include and #define. The preprocessor includes the necessary header files and replaces macros, producing a modified version of the code.

Next, this preprocessed code is given to the **compiler**, which checks the syntax and translates it into **assembly code**, a low-level language closer to machine instructions. The assembly code is then processed by an **assembler**, which converts it into **object code**—a binary format that the computer can understand. However, this object file is not yet ready to run, as it may depend on other code or standard libraries.

To create the final program, a **linker** is used. The linker combines the object file with any required libraries (like stdio.h for input/output functions) and produces an **executable file**. This

file can be run directly on the computer. On Windows, the file typically ends with .exe, while on Linux or macOS, it often has no extension.

Finally, the user can **run the executable file** from the terminal or command line, and the program performs the tasks defined in the source code. This entire process—writing, preprocessing, compiling, assembling, linking, and executing—is what transforms a simple C program into a working application.



Programs:

1. Write a program to demonstrate the library used, default function, variable, multiple data types, and program structure.

CODE:

```

#include <stdio.h> // Standard Input/Output library

// Main function: Entry point of the program

int main() {

    // Variable declarations using multiple data types

    int age = 21;           // Integer variable to store age

    float height = 5.9;     // Float variable to store height in
    feet

    char grade = 'A';       // Character variable to store grade

    double gpa = 3.85;      // Double variable to store GPA with
    higher precision

    char name[] = "Mahendra"; // String (character array) to store
    name

    // Displaying values using default output function printf()

    printf("Student Information:\n");

    printf("Name: %s\n", name);

    printf("Age: %d years\n", age);

    printf("Height: %.1f feet\n", height);

    printf("Grade: %c\n", grade);

    printf("GPA: %.2lf\n", gpa);

    return 0; // Indicates successful termination of the program
}

```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN1.exe
Student Information:
Name: Mahendra
Age: 21 years
Height: 5.9 feet
Grade: A
GPA: 1.85

-----
Process exited after 0.7497 seconds with return value 0
Press any key to continue . . .
```

2. Write a C program that takes user input (integer and float) and displays their sum using appropriate data types. Include escape sequences for formatted output.

CODE:

```
#include <stdio.h>

int main() {
    int a;
    float b, sum;

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

    printf("Enter a float value: ");
    scanf("%f", &b);
```

```

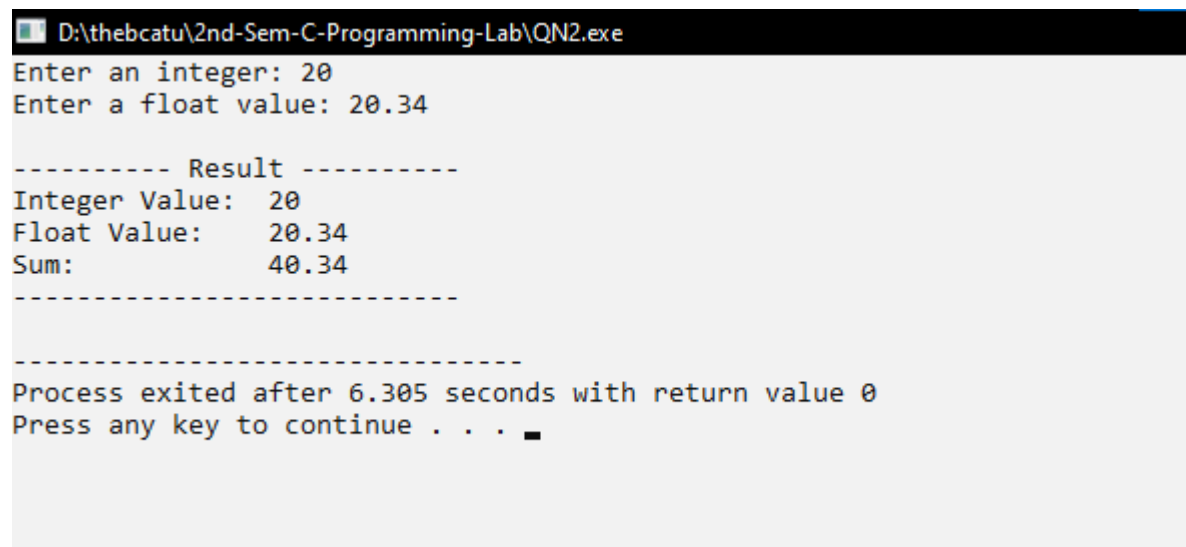
    sum = a + b;

    printf("\n----- Result ----- \n");
    printf("Integer Value:\t%d\n", a);
    printf("Float Value:\t%.2f\n", b);
    printf("Sum:\t\t%.2f\n", sum);
    printf("----- \n");

    return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN2.exe
Enter an integer: 20
Enter a float value: 20.34

----- Result -----
Integer Value:  20
Float Value:    20.34
Sum:            40.34
-----

-----
Process exited after 6.305 seconds with return value 0
Press any key to continue . . .

```

3. Write a C program to demonstrate the use of different tokens (keywords, identifiers, operators, constants) in a simple arithmetic operation.

CODE:

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

```
int main() {
```

```

int a = 10, b = 5, sum, diff, prod;
float div;

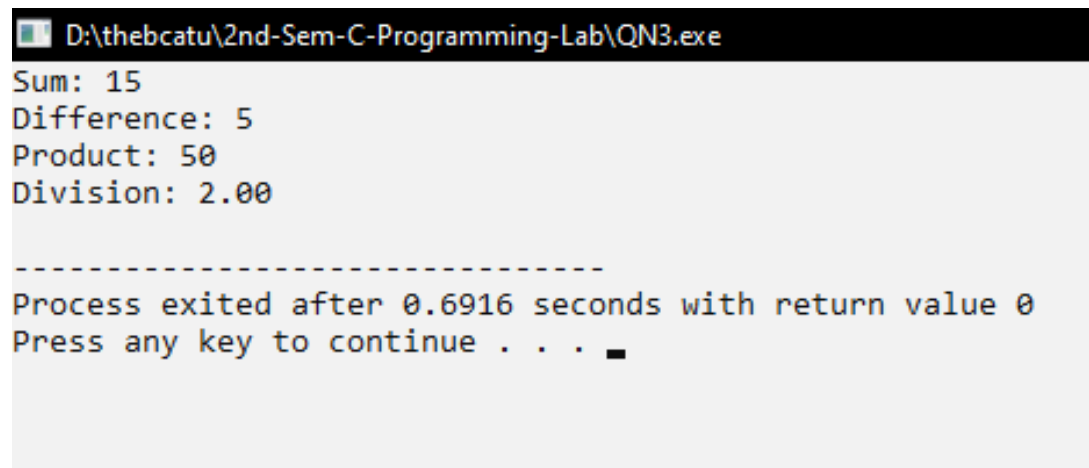
sum = a + b;
diff = a - b;
prod = a * b;
div = (float)a / b;

printf("Sum: %d\n", sum);
printf("Difference: %d\n", diff);
printf("Product: %d\n", prod);
printf("Division: %.2f\n", div);

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN3.exe
Sum: 15
Difference: 5
Product: 50
Division: 2.00

-----
Process exited after 0.6916 seconds with return value 0
Press any key to continue . . .

```

4. Write a C program to showcase preprocessor directives (#define, #ifdef).

CODE:

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



```

#define PI 3.14

#define AREA(radius) (PI * (radius) * (radius))

#define DEBUG

int main() {
    float radius = 5.0;
    float area = AREA(radius);

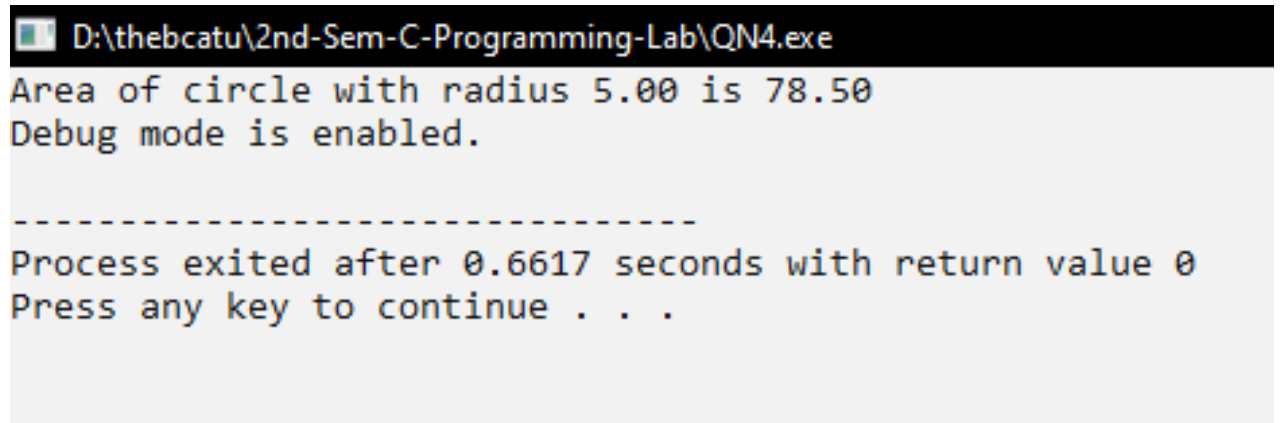
    printf("Area of circle with radius %.2f is %.2f\n", radius,
area);

#ifdef DEBUG
    printf("Debug mode is enabled.\n");
#endif

    return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN4.exe
Area of circle with radius 5.00 is 78.50
Debug mode is enabled.

-----
Process exited after 0.6617 seconds with return value 0
Press any key to continue . . .

```

5. Write a C program to display ASCII values of the given character set (A-Z, a-z, 0-9) using a loop and printf().

CODE:

```
#include <stdio.h>

int main() {
    char ch;

    printf("ASCII values of A-Z:\n");
    for (ch = 'A'; ch <= 'Z'; ch++) {
        printf("%c: %d\n", ch, ch);
    }

    printf("\nASCII values of a-z:\n");
    for (ch = 'a'; ch <= 'z'; ch++) {
        printf("%c: %d\n", ch, ch);
    }

    printf("\nASCII values of 0-9:\n");
    for (ch = '0'; ch <= '9'; ch++) {
        printf("%c: %d\n", ch, ch);
    }

    return 0;
}
```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN5.exe
ASCII values of a-z:
a: 97
b: 98
c: 99
d: 100
e: 101
f: 102
g: 103
h: 104
i: 105
j: 106
k: 107
l: 108
m: 109
n: 110
o: 111
p: 112
q: 113
r: 114
s: 115
t: 116
u: 117
v: 118
w: 119
x: 120
y: 121
z: 122

ASCII values of 0-9:
0: 48
1: 49
2: 50
3: 51
4: 52
5: 53
6: 54
7: 55
8: 56
9: 57

-----
Process exited after 0.722 seconds with return value 0
Press any key to continue . . .
```

6. Write a program that calculates the area and perimeter of a rectangle using arithmetic operators (+, *). Use compound assignment operators (+=, *=) to update the values dynamically.

CODE:

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

```

int main() {
    float length = 5.0, width = 3.0;
    float area, perimeter;

    area = length * width;
    perimeter = 2 * (length + width);

    length += 2.0;
    width += 1.0;

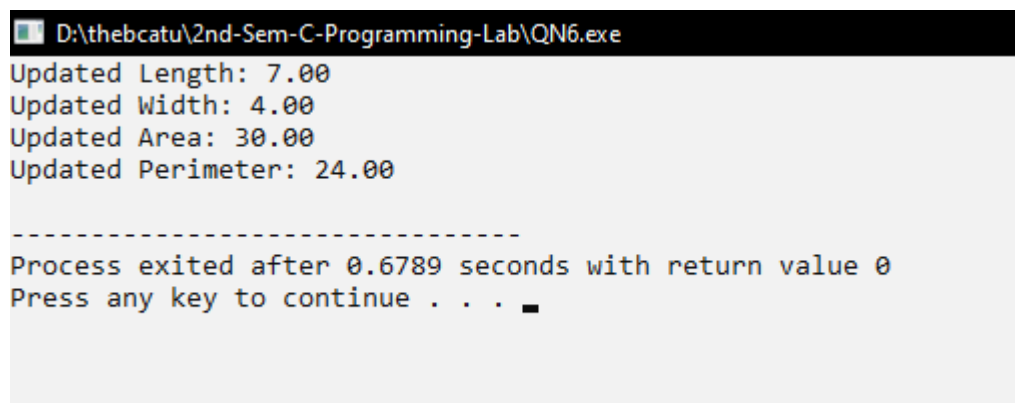
    area *= 2;
    perimeter *= 1.5;

    printf("Updated Length: %.2f\n", length);
    printf("Updated Width: %.2f\n", width);
    printf("Updated Area: %.2f\n", area);
    printf("Updated Perimeter: %.2f\n", perimeter);

    return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN6.exe
Updated Length: 7.00
Updated Width: 4.00
Updated Area: 30.00
Updated Perimeter: 24.00

-----
Process exited after 0.6789 seconds with return value 0
Press any key to continue . . . 

```

7. Write a program to check if a person is eligible for voting (age ≥ 18) AND is a citizen (boolean). Use relational (\geq) and logical operators ($\&\&$).

CODE:

```
#include <stdio.h>

int main() {
    int age;
    int isCitizen;

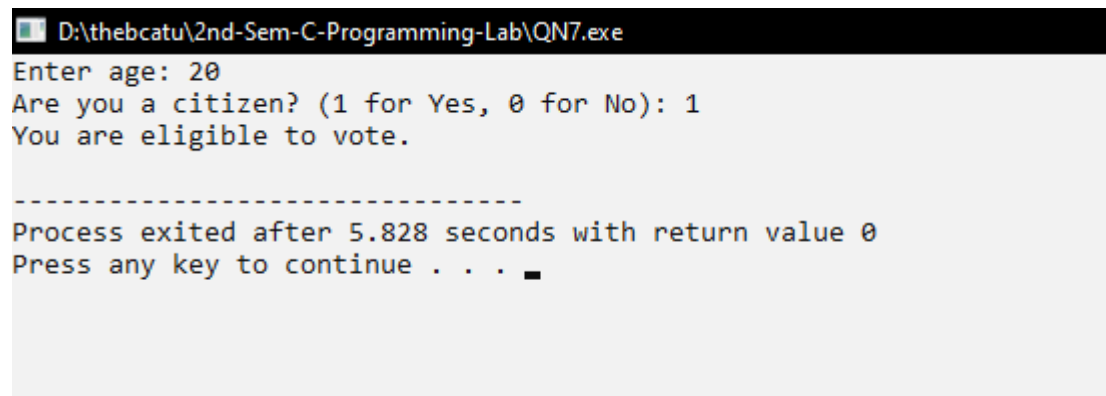
    printf("Enter age: ");
    scanf("%d", &age);

    printf("Are you a citizen? (1 for Yes, 0 for No): ");
    scanf("%d", &isCitizen);

    if (age >= 18 && isCitizen) {
        printf("You are eligible to vote.\n");
    } else {
        printf("You are not eligible to vote.\n");
    }

    return 0;
}
```

OUTPUT:



```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN7.exe
Enter age: 20
Are you a citizen? (1 for Yes, 0 for No): 1
You are eligible to vote.
-----
Process exited after 5.828 seconds with return value 0
Press any key to continue . . . █
```

8. Write a program to swap two numbers using increment/decrement operators and a conditional (ternary) operator to find the larger number.

CODE:

```
#include <stdio.h>

int main() {
    int a, b;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    a = a + 1;
    b = b - 1;

    a = a - 1;
    b = b + 1;

    printf("Swapped values: a = %d, b = %d\n", a, b);

    int larger = (a > b) ? a : b;
    printf("Larger number: %d\n", larger);

    return 0;
}
```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN8.exe
Enter two numbers: 23 21
Swapped values: a = 23, b = 21
Larger number: 23

-----
Process exited after 12.74 seconds with return value 0
Press any key to continue . . .
```

9. Write a program to perform bitwise AND (&) and OR (|) on two integers. Also print their sizes using sizeof.

CODE:

```
#include <stdio.h>

int main() {
    int a, b;

    printf("Enter two integers: ");
    scanf("%d %d", &a, &b);

    printf("Bitwise AND (a & b): %d\n", a & b);
    printf("Bitwise OR (a | b): %d\n", a | b);

    printf("Size of a: %zu bytes\n", sizeof(a));
    printf("Size of b: %zu bytes\n", sizeof(b));
    printf("Size of result (a & b): %zu bytes\n", sizeof(a & b));
    printf("Size of result (a | b): %zu bytes\n", sizeof(a | b));

    return 0;
}
```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN9.exe
Enter two integers: 1243 3276
Bitwise AND (a & b): 1224
Bitwise OR (a | b): 3295
Size of a: 4 bytes
Size of b: 4 bytes
Size of result (a & b): 4 bytes
Size of result (a | b): 4 bytes

-----
Process exited after 17.59 seconds with return value 0
Press any key to continue . . .
```

10. Write a program to evaluate the expression $(a + b) * (c / d)$ with explicit type casting to ensure floating-point division.

CODE:

```
#include <stdio.h>

int main() {
    int a = 5, b = 3, c = 10, d = 4;
    float result;

    result = (a + b) * ((float)c / d);

    printf("Result of (a + b) * (c / d): %.2f\n", result);

    return 0;
}
```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN10.exe
Result of (a + b) * (c / d): 20.00

-----
Process exited after 0.6714 seconds with return value 0
Press any key to continue . . .
```


11. Write a program that takes a user's name (string), age (integer), and height in meters (float) using `scanf()` and displays them back using `printf()` with proper formatting.

CODE:

```
#include <stdio.h>

int main() {
    char name[50];
    int age;
    float height;

    printf("Enter your name: ");
    scanf("%[^\n]", name);

    printf("Enter your age: ");
    scanf("%d", &age);

    printf("Enter your height in meters: ");
    scanf("%f", &height);

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

    return 0;
}
```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN11.exe
Enter your name: Mahendra Mahara
Enter your age: 20
Enter your height in meters: 6

--- User Information ---
Name: Mahendra Mahara
Age: 20
Height: 6.00 meters

-----
Process exited after 13.42 seconds with return value 0
Press any key to continue . . .
```

12. Write a program that reads a single character using `getchar()` and prints it twice using `putchar()`.

CODE:

```
#include <stdio.h>

int main() {
    char ch;

    printf("Enter a character: ");
    ch = getchar();

    printf("The character entered is: ");
    putchar(ch);
    putchar(ch);

    return 0;
}
```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN12.exe
Enter a character: A
The character entered is: AA
-----
Process exited after 3.784 seconds with return value 0
Press any key to continue . . .
```

13. Write a program that uses `getch()` to read a character without echoing it to the screen, then displays it using `putch()`.

CODE:

```
#include <conio.h>

#include <stdio.h>

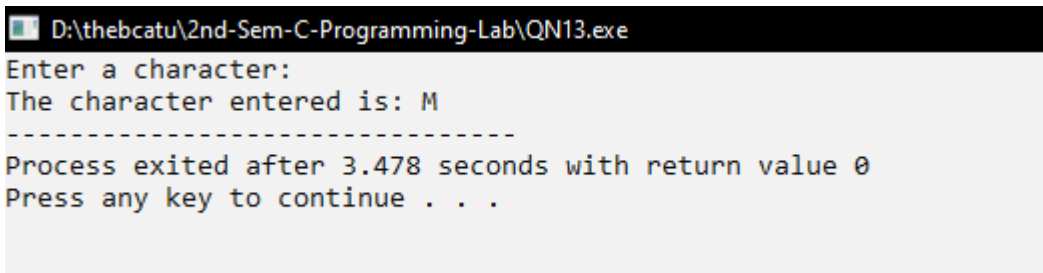
int main() {
    char ch;

    printf("Enter a character: ");
    ch = getch();

    printf("\nThe character entered is: ");
    putch(ch);

    return 0;
}
```

OUTPUT:



```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN13.exe
Enter a character:
The character entered is: M
-----
Process exited after 3.478 seconds with return value 0
Press any key to continue . . .
```

14. Write a program that reads a full line of text (including spaces) using `gets()` and prints it in uppercase using `puts()`.

CODE:

```
#include <stdio.h>

#include <ctype.h>
```

```

int main() {
    char str[100];

    printf("Enter a line of text: ");
    gets(str);

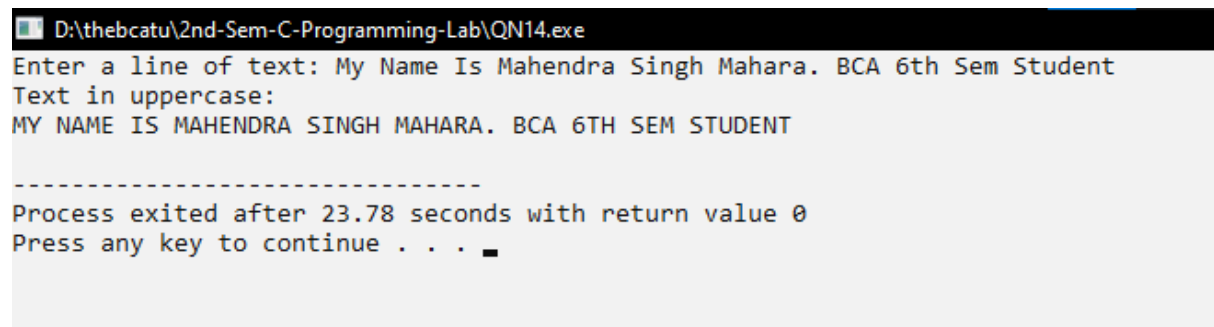
    for (int i = 0; str[i]; i++) {
        str[i] = toupper(str[i]);
    }

    puts("Text in uppercase:");
    puts(str);

    return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN14.exe
Enter a line of text: My Name Is Mahendra Singh Mahara. BCA 6th Sem Student
Text in uppercase:
MY NAME IS MAHENDRA SINGH MAHARA. BCA 6TH SEM STUDENT
-----
Process exited after 23.78 seconds with return value 0
Press any key to continue . . .

```

15. Write a program that uses getch() to read a character (with echo), then prints its ASCII value using printf().

CODE:

```

#include <conio.h>
#include <stdio.h>

```

```

int main() {

```

```

char ch;

printf("Enter a character: ");

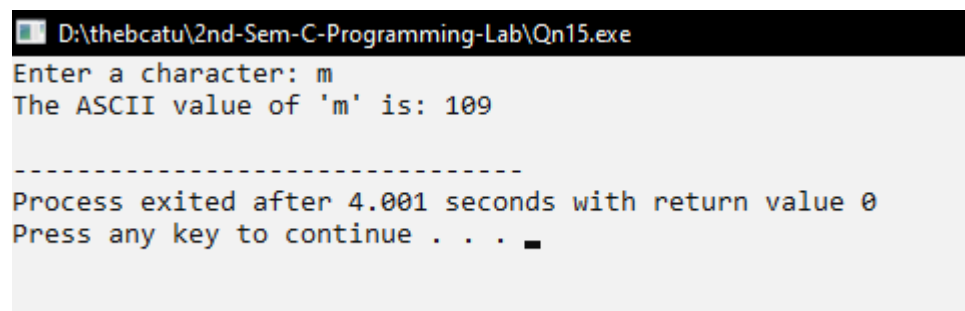
ch = getche();

printf("\nThe ASCII value of '%c' is: %d\n", ch, ch);

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\Qn15.exe
Enter a character: m
The ASCII value of 'm' is: 109
-----
Process exited after 4.001 seconds with return value 0
Press any key to continue . . .

```

16. Write a program to simulate an ATM menu using switch-case for options (withdrawal, deposit, balance check) and if-else to validate the PIN. Use goto to return to the main menu after each transaction.

CODE:

```

#include <stdio.h>

int main() {
    int pin, option;
    float balance = 1000.0, amount;

    printf("Enter your PIN: ");
    scanf("%d", &pin);
}

```

```

    if (pin != 1234) {
        printf("Invalid PIN. Access denied.\n");
        return 0;
    }

menu:

    printf("\n----- ATM Menu ----- \n");
    printf("1. Withdrawal\n");
    printf("2. Deposit\n");
    printf("3. Balance Check\n");
    printf("4. Exit\n");
    printf("Select an option: ");
    scanf("%d", &option);

    switch (option) {
        case 1:
            printf("Enter amount to withdraw: ");
            scanf("%f", &amount);
            if (amount > balance) {
                printf("Insufficient balance.\n");
            } else {
                balance -= amount;
                printf("Withdrawal successful. Remaining balance:
%.2f\n", balance);
            }
            goto menu;

        case 2:
            printf("Enter amount to deposit: ");

```

```
        scanf("%f", &amount);
        balance += amount;
        printf("Deposit successful. New balance: %.2f\n",
balance);
        goto menu;

    case 3:
        printf("Current balance: %.2f\n", balance);
        goto menu;

    case 4:
        printf("Thank you for using the ATM.\n");
        break;

    default:
        printf("Invalid option.\n");
        goto menu;
}

return 0;
}
```

OUTPUT:

```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN16.exe
Enter your PIN: 1234

----- ATM Menu -----
1. Withdrawal
2. Deposit
3. Balance Check
4. Exit
Select an option: 1
Enter amount to withdraw: 2000
Insufficient balance.

----- ATM Menu -----
1. Withdrawal
2. Deposit
3. Balance Check
4. Exit
Select an option: 1
Enter amount to withdraw: 100
Withdrawal successful. Remaining balance: 900.00

----- ATM Menu -----
1. Withdrawal
2. Deposit
3. Balance Check
4. Exit
Select an option: █

```

17. Write a program to print prime numbers between 1 and 100 using a for loop, break to exit early, and continue to skip even numbers (optimization). Add a do-while loop to repeat the process if the user wants.

CODE:

```

#include <stdio.h>

int main() {
    int i, j, isPrime;
    char repeat;

    do {
        printf("Prime numbers between 1 and 100:\n");

```



```

    for (i = 2; i <= 100; i++) {
        if (i != 2 && i % 2 == 0)
            continue;

        isPrime = 1;
        for (j = 2; j * j <= i; j++) {
            if (i % j == 0) {
                isPrime = 0;
                break;
            }
        }

        if (isPrime)
            printf("%d ", i);
    }

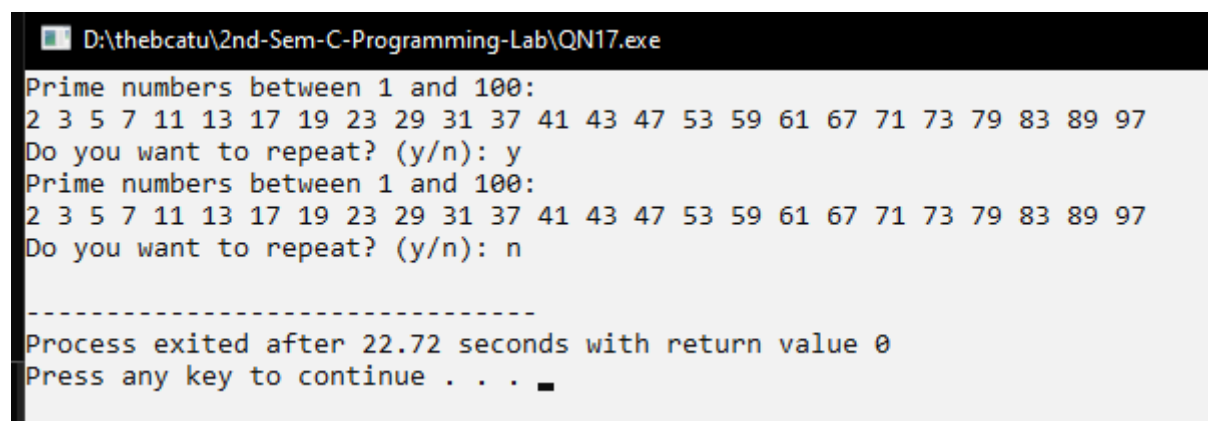
    printf("\nDo you want to repeat? (y/n): ");
    scanf(" %c", &repeat);

} while (repeat == 'y' || repeat == 'Y');

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN17.exe
Prime numbers between 1 and 100:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
Do you want to repeat? (y/n): y
Prime numbers between 1 and 100:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
Do you want to repeat? (y/n): n

-----
Process exited after 22.72 seconds with return value 0
Press any key to continue . . .

```

18. Write a program to print a pyramid pattern using nested for loops. Ask the user to choose between a pyramid of numbers or stars using if-else if.

CODE:

```
#include <stdio.h>

int main() {
    int choice, i, j, rows;

    printf("Choose pattern type:\n");
    printf("1. Number Pyramid\n");
    printf("2. Star Pyramid\n");
    printf("Enter your choice (1 or 2): ");
    scanf("%d", &choice);

    printf("Enter number of rows: ");
    scanf("%d", &rows);

    if (choice == 1) {
        for (i = 1; i <= rows; i++) {
            for (j = 1; j <= rows - i; j++) {
                printf(" ");
            }
            for (j = 1; j <= 2 * i - 1; j++) {
                printf("%d", i);
            }
            printf("\n");
        }
    } else if (choice == 2) {
```

```

        for (i = 1; i <= rows; i++) {
            for (j = 1; j <= rows - i; j++) {
                printf(" ");
            }
            for (j = 1; j <= 2 * i - 1; j++) {
                printf("*");
            }
            printf("\n");
        }
    } else {
        printf("Invalid choice.\n");
    }

    return 0;
}

```

OUTPUT:

```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN18.exe
Choose pattern type:
1. Number Pyramid
2. Star Pyramid
Enter your choice (1 or 2): 2
Enter number of rows: 8
      *
     ***
    *****
   ********
  *********
 *****
  *****
   *****
    *****
     *****

-----
Process exited after 6.29 seconds with return value 0
Press any key to continue . . . █

```

19. Write a program that:

- Takes 10 integers as input into a 1D array.

- Sorts them in ascending order using **Bubble Sort**.
- Re-sorts them in descending order using **Selection Sort**.
- Prints both results.

CODE:

```
#include <stdio.h>

int main() {
    int arr[10], i, j, temp;

    printf("Enter 10 integers:\n");
    for (i = 0; i < 10; i++) {
        scanf("%d", &arr[i]);
    }

    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9 - i; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    printf("\nAscending order (Bubble Sort):\n");
    for (i = 0; i < 10; i++) {
        printf("%d ", arr[i]);
    }
```

```

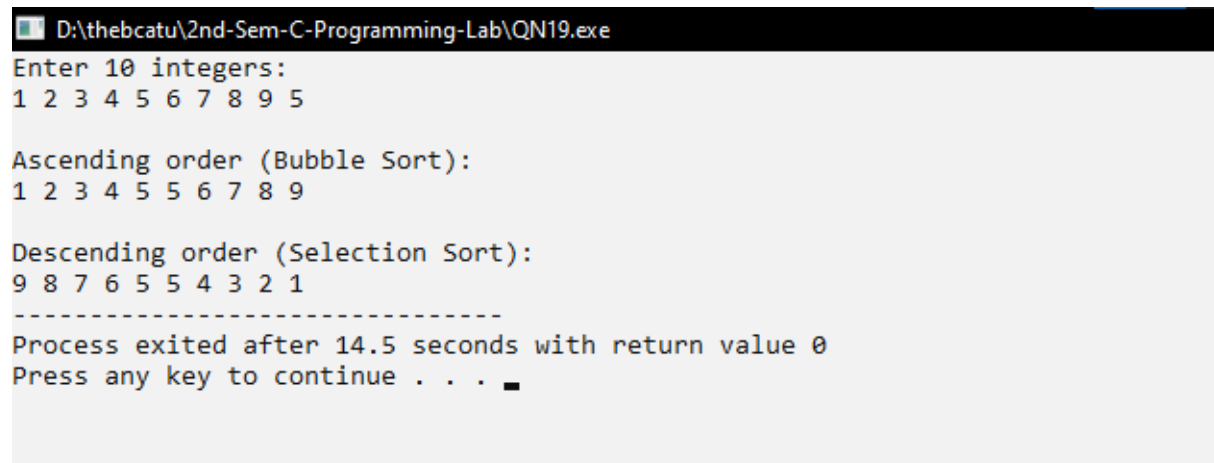
    for (i = 0; i < 9; i++) {
        int max_idx = i;
        for (j = i + 1; j < 10; j++) {
            if (arr[j] > arr[max_idx]) {
                max_idx = j;
            }
        }
        if (max_idx != i) {
            temp = arr[i];
            arr[i] = arr[max_idx];
            arr[max_idx] = temp;
        }
    }

    printf("\n\nDescending order (Selection Sort):\n");
    for (i = 0; i < 10; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}

```

OUTOUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN19.exe
Enter 10 integers:
1 2 3 4 5 6 7 8 9 5

Ascending order (Bubble Sort):
1 2 3 4 5 5 6 7 8 9

Descending order (Selection Sort):
9 8 7 6 5 5 4 3 2 1
-----
Process exited after 14.5 seconds with return value 0
Press any key to continue . . .

```

20. Write a program that:

- Declares a 3x3 matrix (2D array) initialized with values.
- Searches for a user-input value using **Sequential Search**.
- Prints the value's position if found, else "Not Found".

CODE:

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

```
int main() {
    int matrix[3][3] = {
        {5, 8, 2},
        {9, 1, 7},
        {6, 3, 4}
    };
    int i, j, num, found = 0;

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

    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            if (matrix[i][j] == num) {
                printf("Value found at position: [%d][%d]\n", i, j);
                found = 1;
                break;
            }
        }
        if (found)
            break;
    }
}
```

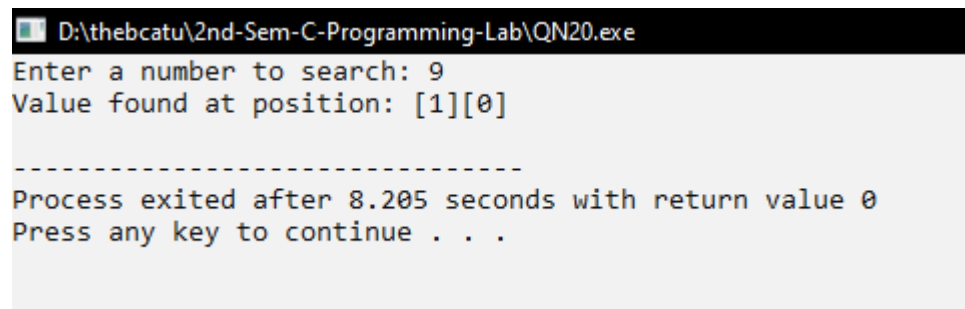
```

    if (!found)
        printf("Not Found\n");

    return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN20.exe
Enter a number to search: 9
Value found at position: [1][0]
-----
Process exited after 8.205 seconds with return value 0
Press any key to continue . . .

```

21. Write a program that:

- Takes a string input (e.g., "Programming").
- Counts vowels and consonants using a character array.
- Reverses the string and prints both results.

CODE:

```

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

int main() {
    char str[100], rev[100];
    int i, len, vowels = 0, consonants = 0;

    printf("Enter a string: ");
    scanf("%s", str);

```

```

len = strlen(str);

for (i = 0; i < len; i++) {
    char ch = tolower(str[i]);
    if (ch >= 'a' && ch <= 'z') {
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' ||
ch == 'u')
            vowels++;
        else
            consonants++;
    }
}

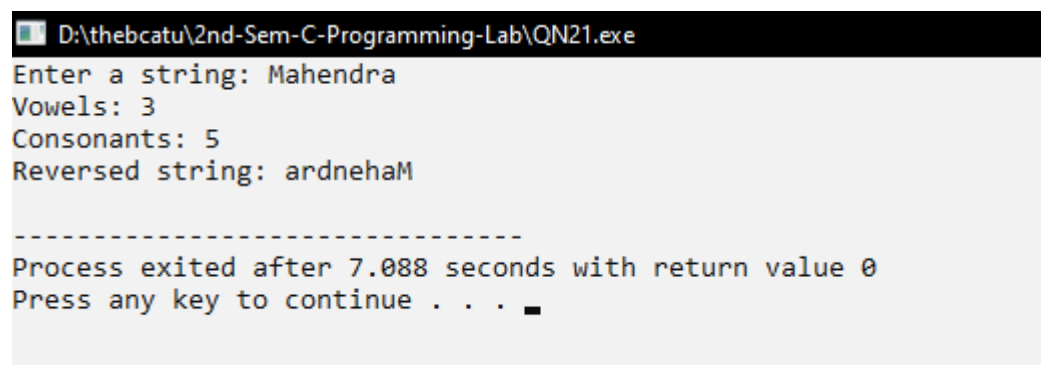
for (i = 0; i < len; i++) {
    rev[i] = str[len - 1 - i];
}
rev[len] = '\0';

printf("Vowels: %d\n", vowels);
printf("Consonants: %d\n", consonants);
printf("Reversed string: %s\n", rev);

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN21.exe
Enter a string: Mahendra
Vowels: 3
Consonants: 5
Reversed string: ardnehaM
-----
Process exited after 7.088 seconds with return value 0
Press any key to continue . . .

```


22. Call by Value vs. Call by Reference & Storage Classes

Write a program with:

- A function `squareByValue` (call by value) that squares an integer.
- A function `squareByReference` (call by reference) that modifies the original variable.
- Use static storage class to count how many times each function is called.

CODE:

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

```
void squareByValue(int n) {  
    static int count1 = 0;  
    count1++;  
    n = n * n;  
    printf("squareByValue result: %d (called %d times)\n", n,  
count1);  
}
```

```
void squareByReference(int *n) {  
    static int count2 = 0;  
    count2++;  
    *n = (*n) * (*n);  
    printf("squareByReference result: %d (called %d times)\n", *n,  
count2);  
}
```

```
int main() {  
    int a, b;
```

```

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

b = a;

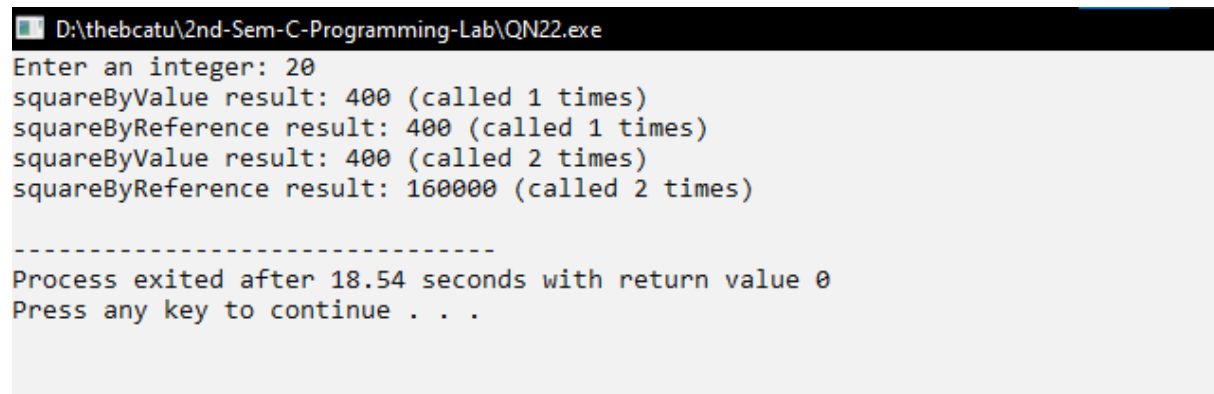
squareByValue(a);
squareByReference(&b);

squareByValue(a);
squareByReference(&b);

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN22.exe
Enter an integer: 20
squareByValue result: 400 (called 1 times)
squareByReference result: 400 (called 1 times)
squareByValue result: 400 (called 2 times)
squareByReference result: 160000 (called 2 times)

-----
Process exited after 18.54 seconds with return value 0
Press any key to continue . . .

```

23. Write a program that:

- Uses recursion to calculate factorial.
- Passes an array to a function to find its sum.
- Passes a string to a function to count vowels.

CODE:

```

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

```

```

int factorial(int n) {
    if (n == 0)
        return 1;
    return n * factorial(n - 1);
}

```

```

int sumArray(int arr[], int size) {
    int sum = 0, i;
    for (i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}

```

```

int countVowels(char str[]) {
    int i, count = 0;
    for (i = 0; str[i]; i++) {
        char ch = tolower(str[i]);
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch
== 'u')
            count++;
    }
    return count;
}

```

```

int main() {
    int num, fact;
    int arr[5], i, sum;

```

```

char str[100];

printf("Enter a number for factorial: ");
scanf("%d", &num);
fact = factorial(num);
printf("Factorial of %d = %d\n", num, fact);

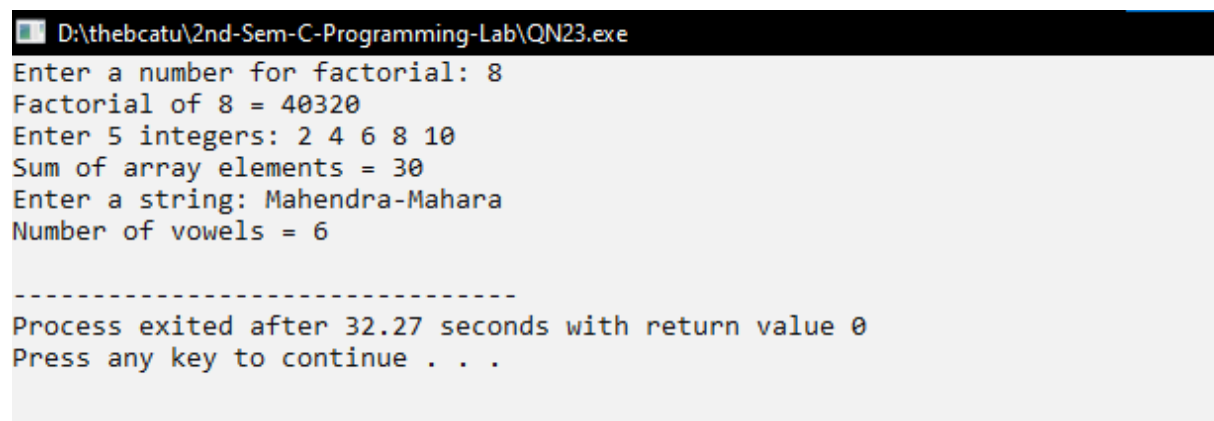
printf("Enter 5 integers: ");
for (i = 0; i < 5; i++) {
    scanf("%d", &arr[i]);
}
sum = sumArray(arr, 5);
printf("Sum of array elements = %d\n", sum);

printf("Enter a string: ");
scanf("%s", str);
printf("Number of vowels = %d\n", countVowels(str));

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN23.exe
Enter a number for factorial: 8
Factorial of 8 = 40320
Enter 5 integers: 2 4 6 8 10
Sum of array elements = 30
Enter a string: Mahendra-Mahara
Number of vowels = 6

-----
Process exited after 32.27 seconds with return value 0
Press any key to continue . . .

```

24. Write a program that:

- Defines a macro SQUARE(x) to compute the square of a number.
- Uses a function with no arguments/return to print a greeting.
- Uses a function with default arguments (simulated via overloading in C).

CODE:

```
#include <stdio.h>

#define SQUARE(x) ((x) * (x))

void greet() {
    printf("Hello! Welcome to C programming.\n");
}

void displayValues(int a, int b) {
    printf("Values: a = %d, b = %d\n", a, b);
}

void displayDefault() {
    displayValues(10, 20);
}

int main() {
    int num;

    greet();

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

```

printf("Square of %d is %d\n", num, SQUARE(num));

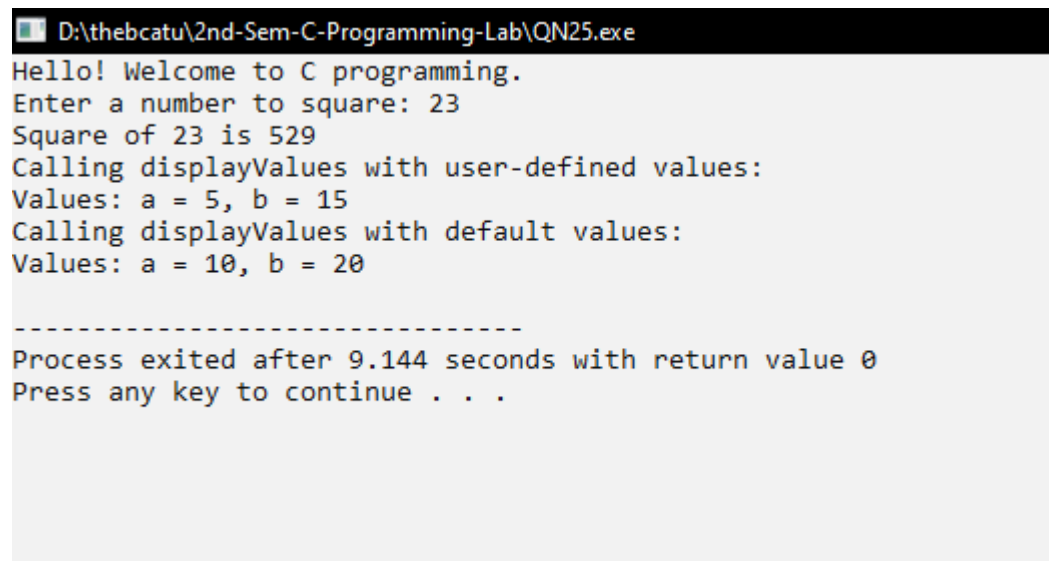
printf("Calling displayValues with user-defined values:\n");
displayValues(5, 15);

printf("Calling displayValues with default values:\n");
displayDefault();

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN25.exe
Hello! Welcome to C programming.
Enter a number to square: 23
Square of 23 is 529
Calling displayValues with user-defined values:
Values: a = 5, b = 15
Calling displayValues with default values:
Values: a = 10, b = 20

-----
Process exited after 9.144 seconds with return value 0
Press any key to continue . . .

```

25. Write a program that:

- Declares an integer array and a pointer to traverse it.
- Uses pointer arithmetic (++ , --) to find the sum and reverse the array.
- Demonstrates & (address-of) and * (dereference) operators.

CODE:

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

```
int main() {
```

```

int arr[5] = {1, 2, 3, 4, 5};
int *ptr = arr;
int sum = 0;
int i;

for (i = 0; i < 5; i++) {
    sum += *ptr;
    ptr++;
}

printf("Sum: %d\n", sum);

ptr = arr + 4;
printf("Reversed array: ");
for (i = 0; i < 5; i++) {
    printf("%d ", *ptr);
    ptr--;
}
printf("\n");

int x = 10;
int *p = &x;
printf("Address of x: %p\n", (void*)&x);
printf("Value using dereference: %d\n", *p);

return 0;
}

```

OUTPUT:

```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN24.exe
Sum: 15
Reversed array: 5 4 3 2 1
Address of x: 000000000062FDEC
Value using dereference: 10

-----
Process exited after 0.758 seconds with return value 0
Press any key to continue . . .
```

26. Write a program that:

- Passes a pointer to a function to modify an array.
- Uses an array of pointers to sort strings.
- Includes a pointer-to-pointer example.

CODE:

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

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

```
void modifyArray(int *arr, int size) {
    int i;
    for (i = 0; i < size; i++) {
        *(arr + i) *= 2;
    }
}
```

```
void sortStrings(char *str[], int n) {
    int i, j;
    char *temp;
    for (i = 0; i < n - 1; i++) {
```



```

        for (j = i + 1; j < n; j++) {
            if (strcmp(str[i], str[j]) > 0) {
                temp = str[i];
                str[i] = str[j];
                str[j] = temp;
            }
        }
    }
}

```

```

int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    int i;
    modifyArray(arr, 5);
    printf("Modified array: ");
    for (i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    char *names[] = {"Zara", "Alex", "John", "Bob"};
    int n = 4;
    sortStrings(names, n);
    printf("Sorted names: ");
    for (i = 0; i < n; i++) {
        printf("%s ", names[i]);
    }
    printf("\n");
}

```

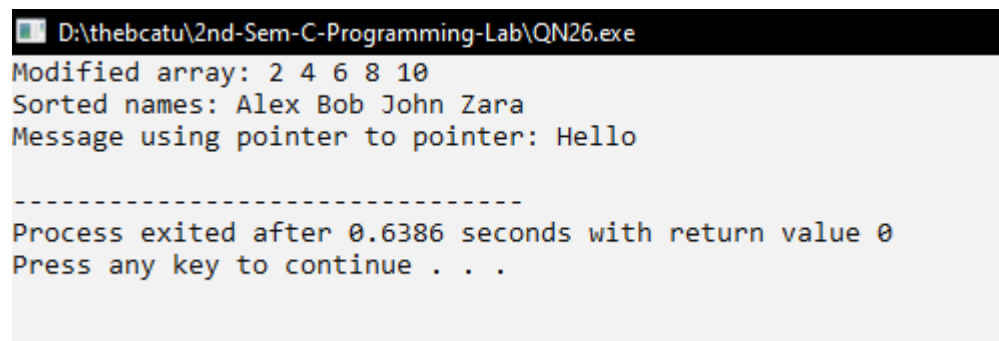
```

char *msg = "Hello";
char **pp = &msg;
printf("Message using pointer to pointer: %s\n", *pp);

return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN26.exe
Modified array: 2 4 6 8 10
Sorted names: Alex Bob John Zara
Message using pointer to pointer: Hello
-----
Process exited after 0.6386 seconds with return value 0
Press any key to continue . . .

```

27. Write a program that:

- Dynamically allocates memory for an integer array using malloc.
- Reads a string into dynamically allocated memory (calloc) and converts it to uppercase.
- Frees memory and checks for allocation errors.

CODE:

```

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

int main() {
    int *arr;
    int n, i;

    printf("Enter number of integers: ");

```

```

scanf("%d", &n);

arr = (int *)malloc(n * sizeof(int));
if (arr == NULL) {
    printf("Memory allocation failed for integers.\n");
    return 1;
}

printf("Enter %d integers:\n", n);
for (i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
}

printf("You entered: ");
for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
}
printf("\n");
free(arr);

char *str;
int length;

printf("Enter length of string: ");
scanf("%d", &length);
getchar();

str = (char *)calloc(length + 1, sizeof(char));
if (str == NULL) {

```

```

        printf("Memory allocation failed for string.\n");
        return 1;
    }

    printf("Enter a string: ");
    fgets(str, length + 1, stdin);

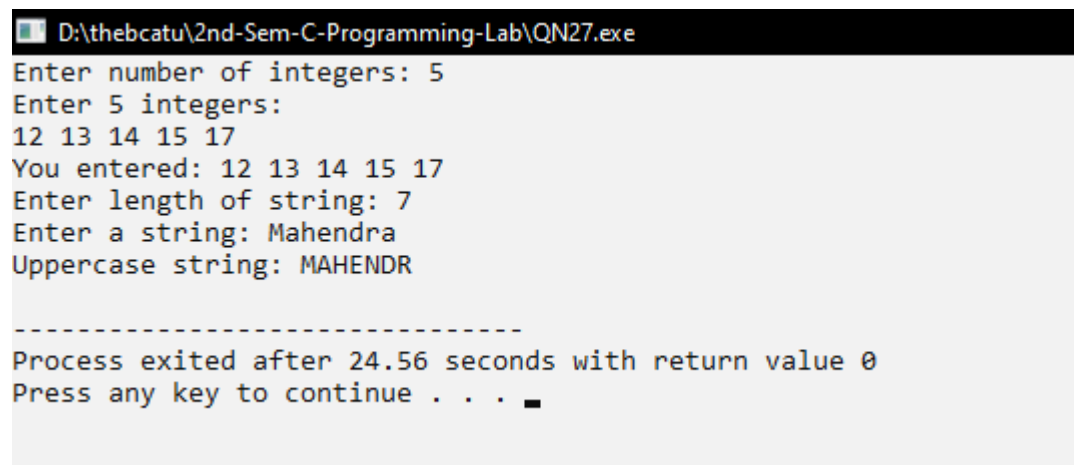
    for (i = 0; str[i] != '\0'; i++) {
        str[i] = toupper(str[i]);
    }

    printf("Uppercase string: %s\n", str);
    free(str);

    return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN27.exe
Enter number of integers: 5
Enter 5 integers:
12 13 14 15 17
You entered: 12 13 14 15 17
Enter length of string: 7
Enter a string: Mahendra
Uppercase string: MAHENDR

-----
Process exited after 24.56 seconds with return value 0
Press any key to continue . . .

```

28. Write a program to:

- Define a structure Student with fields: name, roll, and marks.
- Create an array of 3 Student structures and initialize them.

- Pass the array to a function to print students with marks > 50.

CODE:

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

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

```
struct Student {  
    char name[50];  
    int roll;  
    float marks;  
};
```

```
void printPassed(struct Student s[], int size) {  
    int i;  
    printf("Students with marks > 50:\n");  
    for (i = 0; i < size; i++) {  
        if (s[i].marks > 50) {  
            printf("Name: %s, Roll: %d, Marks: %.2f\n", s[i].name,  
s[i].roll, s[i].marks);  
        }  
    }  
}
```

```
int main() {  
    struct Student students[3] = {  
        {"Mahendra", 1, 75.5},  
        {"Mahara", 2, 45.0},  
        {"Sita", 3, 88.0}  
    };  
}
```

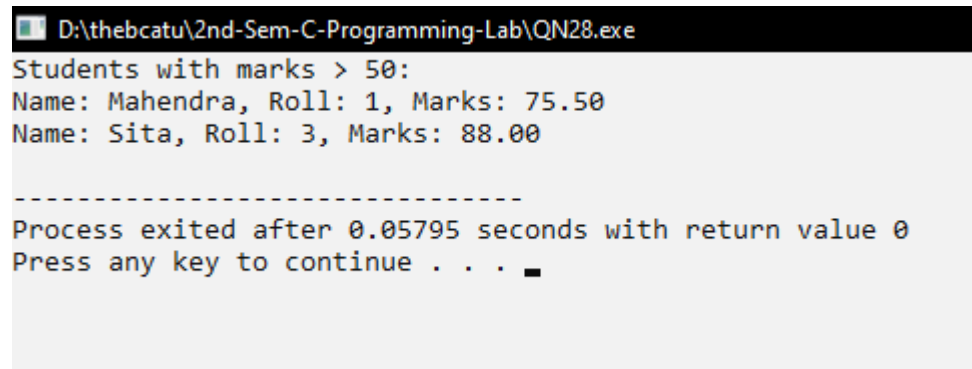
```

    printPassed(students, 3);

    return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN28.exe
Students with marks > 50:
Name: Mahendra, Roll: 1, Marks: 75.50
Name: Sita, Roll: 3, Marks: 88.00
-----
Process exited after 0.05795 seconds with return value 0
Press any key to continue . . .

```

29. Write a program to:

- Define a nested structure Address (city, pin) within Employee.
- Use a pointer to an Employee structure to modify and print its data.
- Include an array within the structure (e.g., skills as a string array).

CODE:

```

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

struct Address {
    char city[50];
    int pin;
};

struct Employee {
    char name[50];

```

```

    int id;
    struct Address address;
    char skills[3][30];
};

int main() {
    struct Employee emp;
    struct Employee *ptr = &emp;

    strcpy(ptr->name, "Kiran");
    ptr->id = 101;
    strcpy(ptr->address.city, "Kathmandu");
    ptr->address.pin = 44600;

    strcpy(ptr->skills[0], "C Programming");
    strcpy(ptr->skills[1], "Data Structures");
    strcpy(ptr->skills[2], "Algorithms");

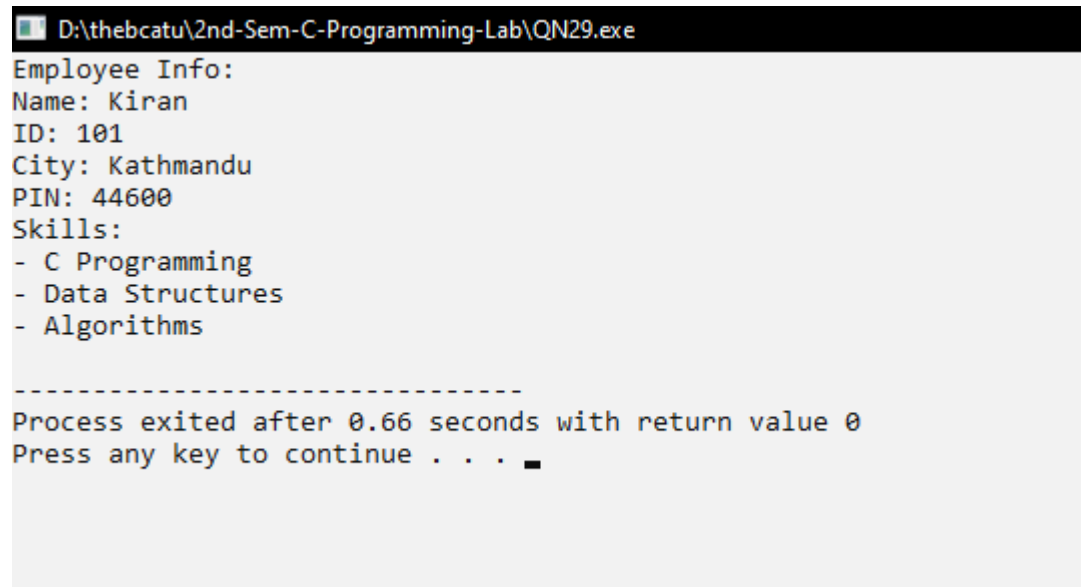
    printf("Employee Info:\n");
    printf("Name: %s\n", ptr->name);
    printf("ID: %d\n", ptr->id);
    printf("City: %s\n", ptr->address.city);
    printf("PIN: %d\n", ptr->address.pin);
    printf("Skills:\n");

    int i;
    for (i = 0; i < 3; i++) {
        printf("- %s\n", ptr->skills[i]);
    }
}

```

```
    return 0;
}
```

OUTPUT:



```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN29.exe
Employee Info:
Name: Kiran
ID: 101
City: Kathmandu
PIN: 44600
Skills:
- C Programming
- Data Structures
- Algorithms

-----
Process exited after 0.66 seconds with return value 0
Press any key to continue . . .
```

30. Write a program to:

- Use a union to store either an integer or a float (demonstrate shared memory).
- Define a structure with bit-fields to represent a 8-bit register (e.g., status: 1, mode: 2 bits).
- Compare sizes of structure and union.

CODE:

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

```
union Data {
    int i;
    float f;
};
```



```

struct Register {
    unsigned int status : 1;
    unsigned int mode : 2;
    unsigned int reserved : 5;
};

int main() {
    union Data d;
    d.i = 10;
    printf("Using union - Integer: %d\n", d.i);
    d.f = 5.5;
    printf("Using union - Float: %.2f\n", d.f);
    printf("After assigning float, integer becomes: %d (shared
memory)\n", d.i);

    struct Register reg;
    reg.status = 1;
    reg.mode = 2;

    printf("Register - Status: %u, Mode: %u\n", reg.status,
reg.mode);

    printf("Size of union: %lu bytes\n", sizeof(d));
    printf("Size of structure with bit-fields: %lu bytes\n",
sizeof(reg));

    return 0;
}

```

OUTPUT:

```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN30.exe
Using union - Integer: 10
Using union - Float: 5.50
After assigning float, integer becomes: 1085276160 (shared memory)
Register - Status: 1, Mode: 2
Size of union: 4 bytes
Size of structure with bit-fields: 4 bytes

-----
Process exited after 1.093 seconds with return value 0
Press any key to continue . . . █

```

31. Write a program to:

- Create a file students.txt and write 3 student records (name, roll, marks) using fprintf().
- Read the file and display records with marks > 60 using fscanf().

CODE:

```

#include <stdio.h>

int main() {
    FILE *fp;
    char name[50];
    int roll;
    float marks;

    fp = fopen("students.txt", "w");
    if (fp == NULL) {
        printf("Error creating file.\n");
        return 1;
    }

    fprintf(fp, "Mahendra 1 75.5\n");
    fprintf(fp, "Mahara 2 45.0\n");
}

```

```

fprintf(fp, "Sita 3 88.0\n");
fclose(fp);



fp = fopen("students.txt", "r");
if (fp == NULL) {
    printf("Error opening file.\n");
    return 1;
}

printf("Students with marks > 60:\n");
while (fscanf(fp, "%s %d %f", name, &roll, &marks) == 3) {
    if (marks > 60) {
        printf("Name: %s, Roll: %d, Marks: %.2f\n", name, roll,
marks);
    }
}

fclose(fp);
return 0;
}

```

OUTPUT:

 students	4/20/2025 4:45 PM	Text Document	1 KB
 mahendra	4/20/2025 4:47 PM	Text Document	1 KB

```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN31.exe
Students with marks > 60:
Name: Mahendra, Roll: 1, Marks: 75.50
Name: Sita, Roll: 3, Marks: 88.00
-----
Process exited after 1.016 seconds with return value 0
Press any key to continue . . .

```

32. Write a program to:

- Update a specific record (e.g., change Bob's marks to 65.0) in students.txt using fseek().
- Append a new student record using fopen() in append mode ("a").

CODE:

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

struct Student {
    char name[50];
    int roll;
    float marks;
};

int main() {
    FILE *fp;
    struct Student s[3];
    int i;

    fp = fopen("students.txt", "r+");
    if (fp == NULL) {
        fp = fopen("students.txt", "w+");
        if (fp == NULL) {
            printf("Error opening file.\n");
            return 1;
        }
        printf("File created.\n");
    }
}
```

```

    for (i = 0; i < 3; i++) {
        fscanf(fp, "%s %d %f", s[i].name, &s[i].roll, &s[i].marks);
    }

    for (i = 0; i < 3; i++) {
        if (strcmp(s[i].name, "Bob") == 0) {
            s[i].marks = 65.0;
            fseek(fp, i * sizeof(struct Student), SEEK_SET);
            fprintf(fp, "%s %d %.1f\n", s[i].name, s[i].roll,
s[i].marks);
            break;
        }
    }

    fclose(fp);

    fp = fopen("students.txt", "a");
    if (fp == NULL) {
        printf("Error opening file for appending.\n");
        return 1;
    }

    fprintf(fp, "Kiran 4 78.5\n");
    fclose(fp);

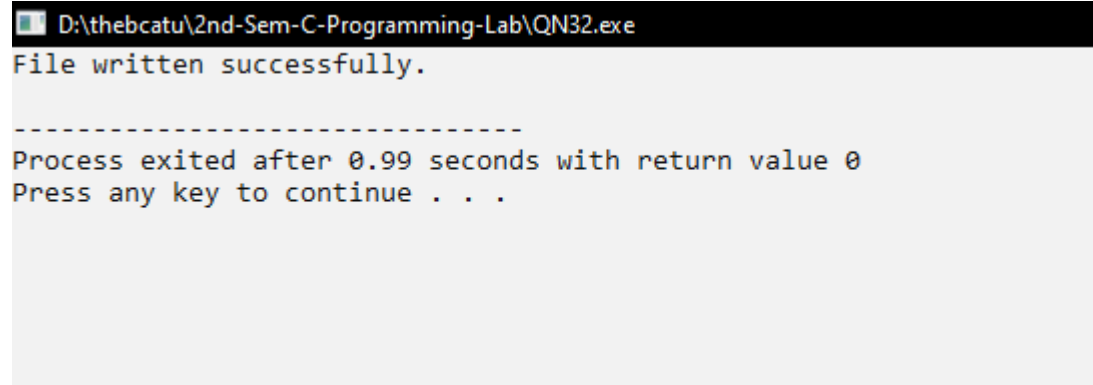
    printf("File written successfully.\n");

    return 0;

```

```
}
```

OUTPUT:



```
D:\thebcatu\2nd-Sem-C-Programming-Lab\QN32.exe
File written successfully.
-----
Process exited after 0.99 seconds with return value 0
Press any key to continue . . .
```

33. Write a program to:

- Write an array of integers to a binary file data.bin using fwrite().
- Read the binary file and print the content in reverse order using fseek() and ftell().

CODE:

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

```
int main() {
    FILE *fp;
    int arr[] = {10, 20, 30, 40, 50};
    int i, size = sizeof(arr) / sizeof(arr[0]);

    fp = fopen("data.bin", "wb");
    if (fp == NULL) {
        printf("Error opening file for writing.\n");
        return 1;
    }
    fwrite(arr, sizeof(int), size, fp);
    fclose(fp);
}
```

```

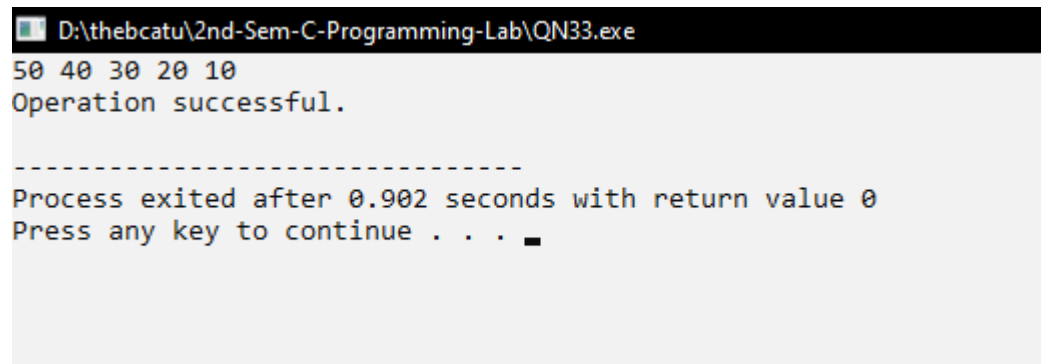
fp = fopen("data.bin", "rb");
if (fp == NULL) {
    printf("Error opening file for reading.\n");
    return 1;
}

fseek(fp, 0, SEEK_END);
long fileSize = ftell(fp);
for (i = size - 1; i >= 0; i--) {
    fseek(fp, i * sizeof(int), SEEK_SET);
    int value;
    fread(&value, sizeof(int), 1, fp);
    printf("%d ", value);
}
printf("\n");

fclose(fp);
printf("Operation successful.\n");
return 0;
}

```

OUTPUT:



```

D:\thebcatu\2nd-Sem-C-Programming-Lab\QN33.exe
50 40 30 20 10
Operation successful.
-----
Process exited after 0.902 seconds with return value 0
Press any key to continue . . .

```

34. Write a program to:

- Initialize graphics mode.
- Draw a **red circle**, **blue rectangle**, and **green polygon** on screen.
- Close graphics mode gracefully.

CODE:

```
#include <graphics.h>
```

```
#include <conio.h>
```

```
int main() {
```

```
    int gd = DETECT, gm;
```

```
    initgraph(&gd, &gm, "");
```

```
    setcolor(RED);
```

```
    circle(320, 240, 100);
```

```
    setcolor(BLUE);
```

```
    rectangle(100, 100, 500, 300);
```

```
    setcolor(GREEN);
```

```
    int points[] = {400, 100, 500, 200, 300, 200};
```

```
    fillpoly(3, points);
```

```
    getch();
```

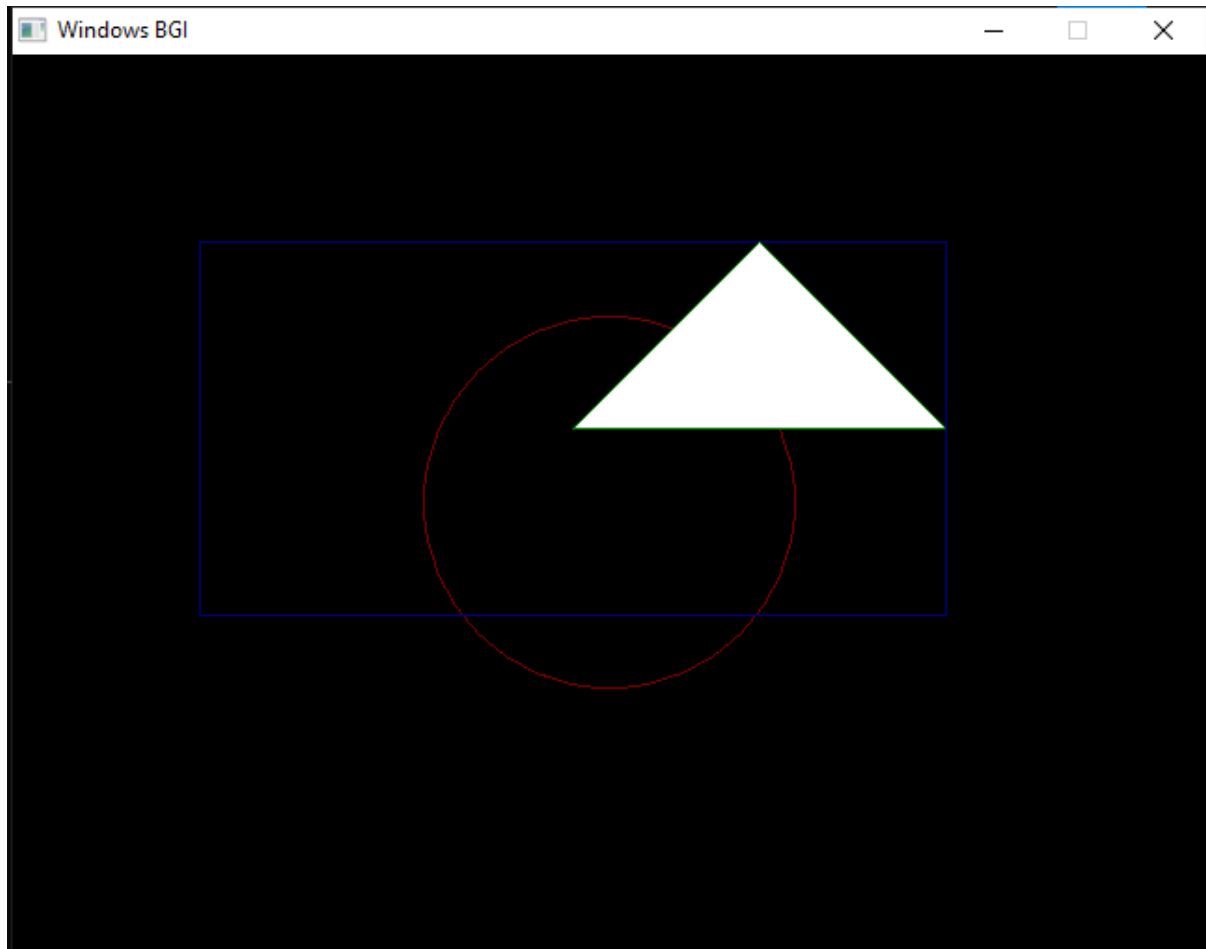
```
    closegraph();
```

```
    return 0;
```



```
}
```

OUTPUT:



35. Write a program to:

- Set background color to **light gray**.
- Print "**Mahendra**" at the center of the screen in **bold font** and **purple color**.

CODE:

```
#include <graphics.h>
```

```
#include <conio.h>
```

```
int main() {
```

```
    int gd = DETECT, gm;
```

```

initgraph(&gd, &gm, "");

setbkcolor(CYAN);
cleardevice();

setcolor(YELLOW);
settextstyle(BOLD_FONT, HORIZ_DIR, 3);
outtextxy(250, 200, "Mahendra");

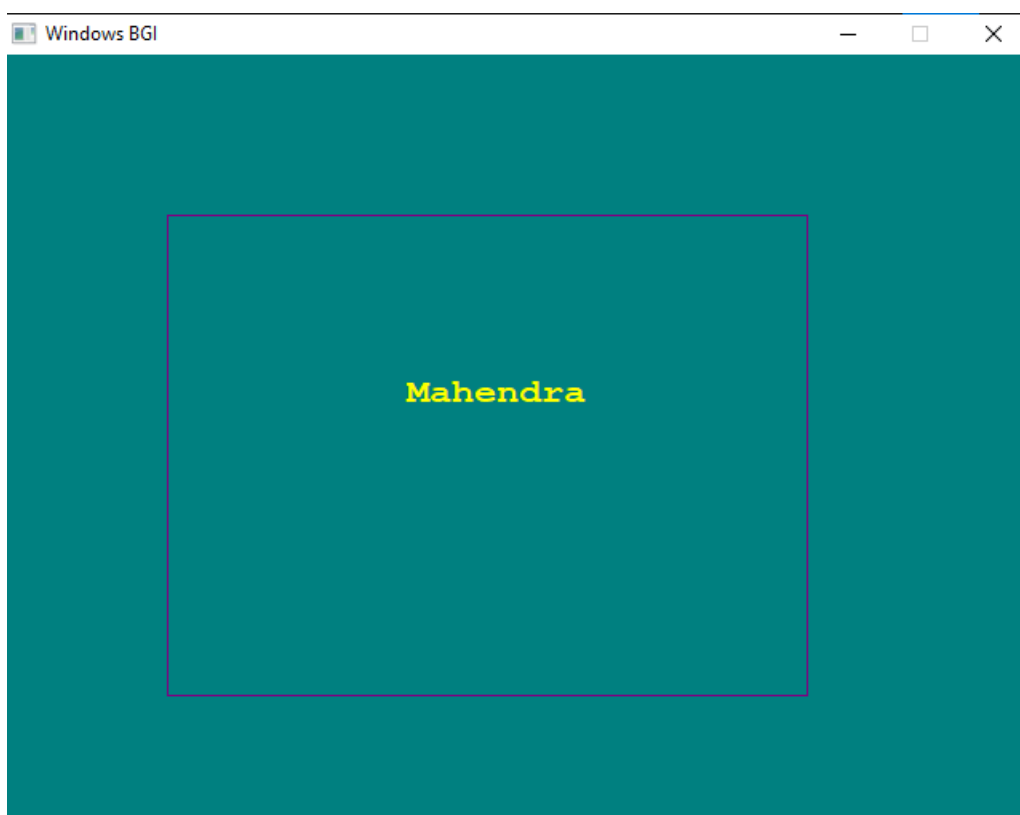
setcolor(MAGENTA);
rectangle(100, 100, 500, 400);

getch();
closegraph();

return 0;
}

```

OUTPUT:



36. Write a program to:

- Draw a simple car (rectangle + circles) that moves horizontally.
- Label it with "Mahendra" above the car.
- Use delay() to animate.

CODE:

```
#include <graphics.h>
```

```
#include <conio.h>
```

```
int main() {
```

```
    int gd = DETECT, gm;
```

```
    int x = 0, y = 300;
```

```
    int width = 200, height = 60;
```

```
    initgraph(&gd, &gm, "");
```

```
    while (!kbhit()) {
```

```
        cleardevice();
```

```
        setcolor(RED);
```

```
        rectangle(x, y, x + width, y + height);
```

```
        setcolor(BLUE);
```

```
        circle(x + 50, y + height, 20);
```

```
        circle(x + 150, y + height, 20);
```

```
        setcolor(YELLOW);
```

```
        settextstyle(BOLD_FONT, HORIZ_DIR, 3);
```

```
        outtextxy(x + 70, y - 30, "Mahendra");
```

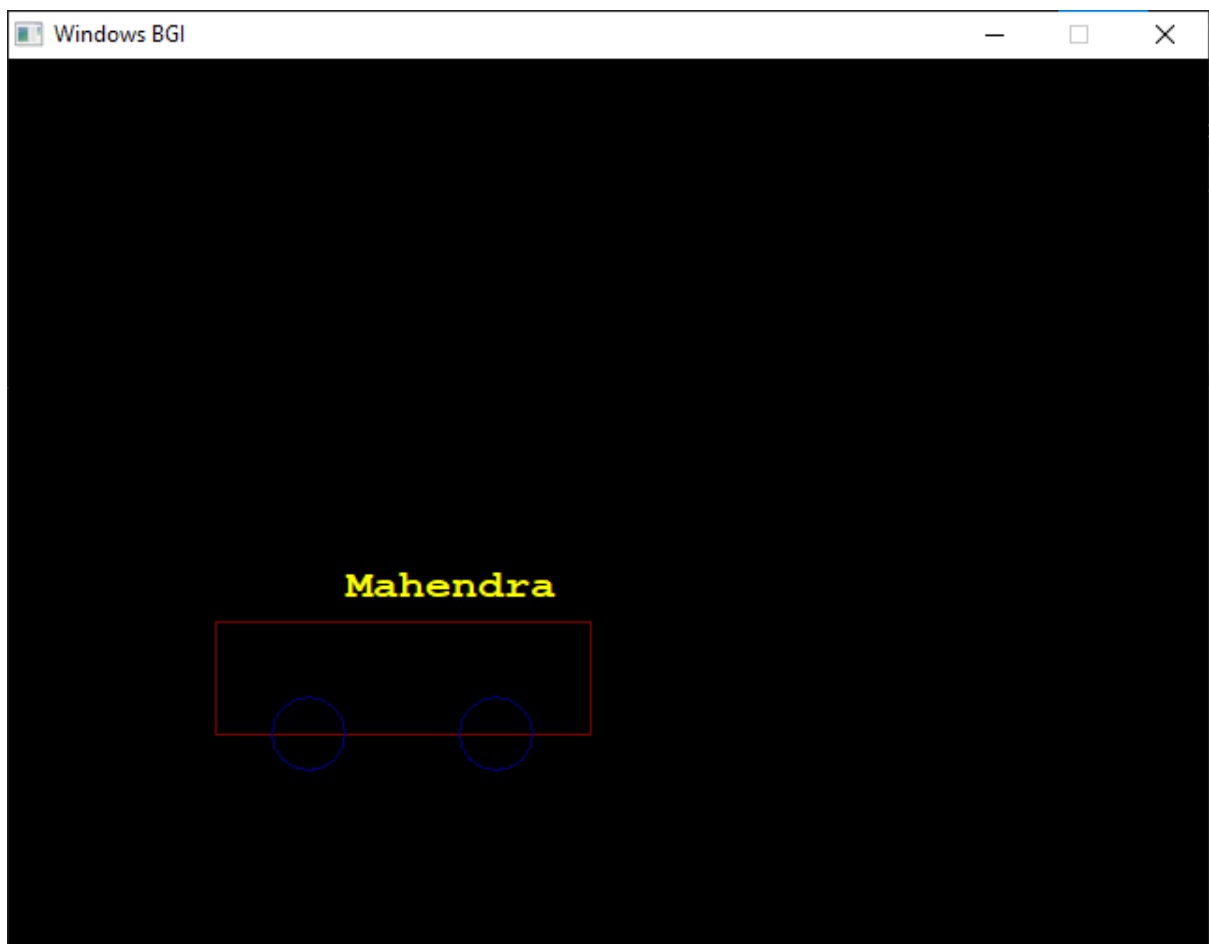
```
    delay(50);

    x += 5;

    if (x > getmaxx()) {
        x = -width;
    }
}

closegraph();
return 0;
}
```

OUTPUT:



37. Write a program to:

- Draw a **pie chart** with 3 slices (RED, BLUE, GREEN).
- Display "Mahendra's Pie Chart" as the title.

CODE:

```
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    int centerX = 320, centerY = 240;
    int radius = 100;
    int angle1 = 120, angle2 = 90, angle3 = 150;

    initgraph(&gd, &gm, "");

    setcolor(RED);
    pieslice(centerX, centerY, 0, angle1, radius); // First slice
    (RED)

    setcolor(BLUE);
    pieslice(centerX, centerY, angle1, angle1 + angle2, radius); //
    Second slice (BLUE)

    setcolor(GREEN);
    pieslice(centerX, centerY, angle1 + angle2, angle1 + angle2 +
    angle3, radius); // Third slice (GREEN)

    setcolor(WHITE);
    setttextstyle(BOLD_FONT, HORIZ_DIR, 3);
```

```

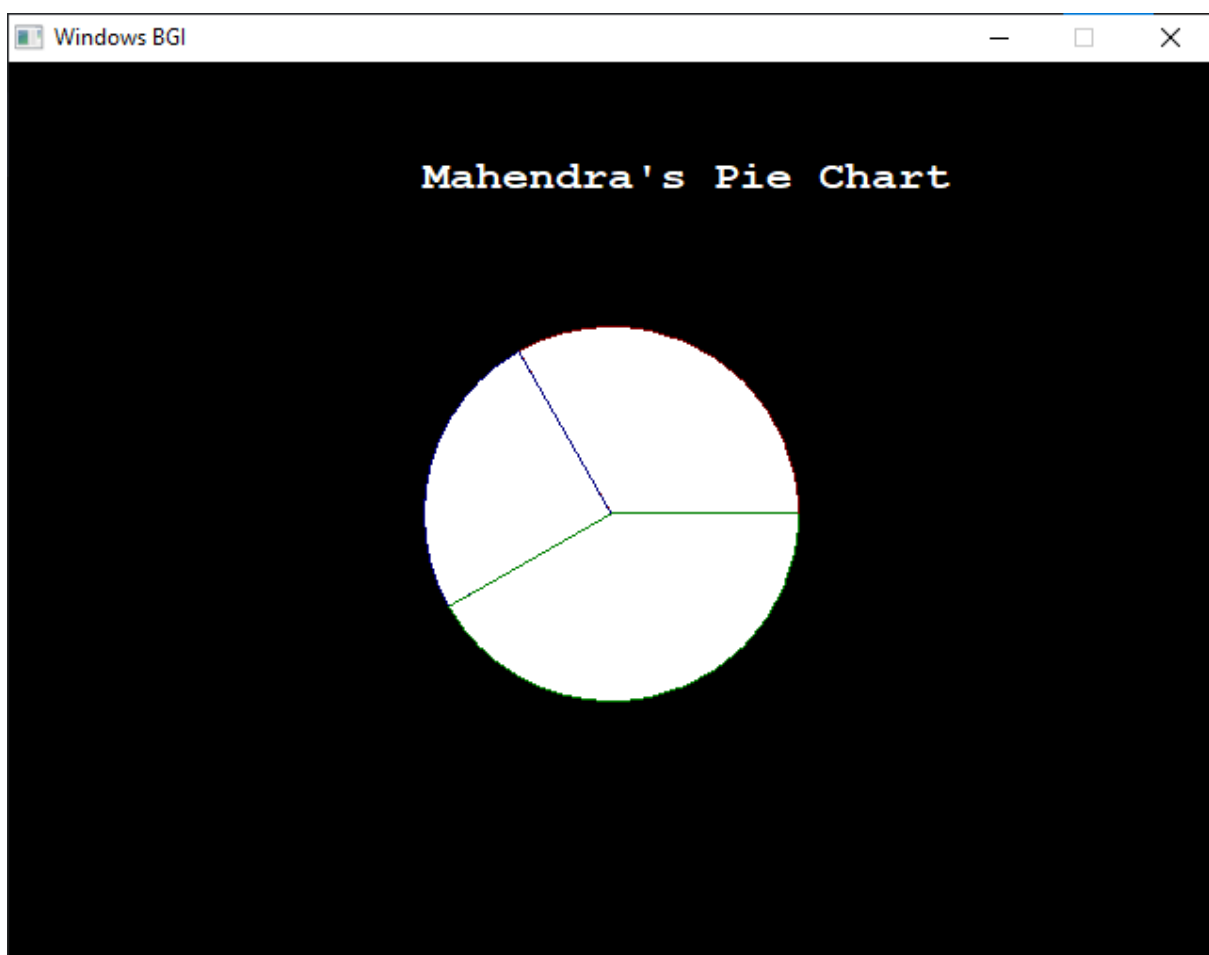
    outtextxy(220, 50, "Mahendra's Pie Chart");

    getch();
    closegraph();

    return 0;
}

```

OUTPUT:



38. Write a program to:

- Display a banner "Mahendra's Graphics Lab" at the top.
- Let the user choose to draw a **circle**, **rectangle**, or **line** by pressing keys (C/R/L).

CODE:

```
#include <graphics.h>
```

```

#include <conio.h>

int main() {
    int gd = DETECT, gm;
    char choice;
    int x = 320, y = 240, radius = 50;
    int left = 100, top = 100, right = 500, bottom = 300;

    initgraph(&gd, &gm, "");

    setcolor(WHITE);
    settextstyle(BOLD_FONT, HORIZ_DIR, 3);
    outtextxy(100, 20, "Mahendra's Graphics Lab");

    while (1) {
        setcolor(WHITE);
        rectangle(0, 40, getmaxx(), getmaxy());
        setcolor(BLACK);

        outtextxy(100, 60, "Choose shape to draw: ");
        outtextxy(100, 90, "Press 'C' for Circle, 'R' for Rectangle,
'L' for Line");

        choice = getch();

        if (choice == 'C' || choice == 'c') {
            setcolor(RED);
            circle(x, y, radius);
        } else if (choice == 'R' || choice == 'r') {

```

```

        setcolor(BLUE);
        rectangle(left, top, right, bottom);
    } else if (choice == 'L' || choice == 'l') {
        setcolor(GREEN);
        line(left, top, right, bottom);
    }

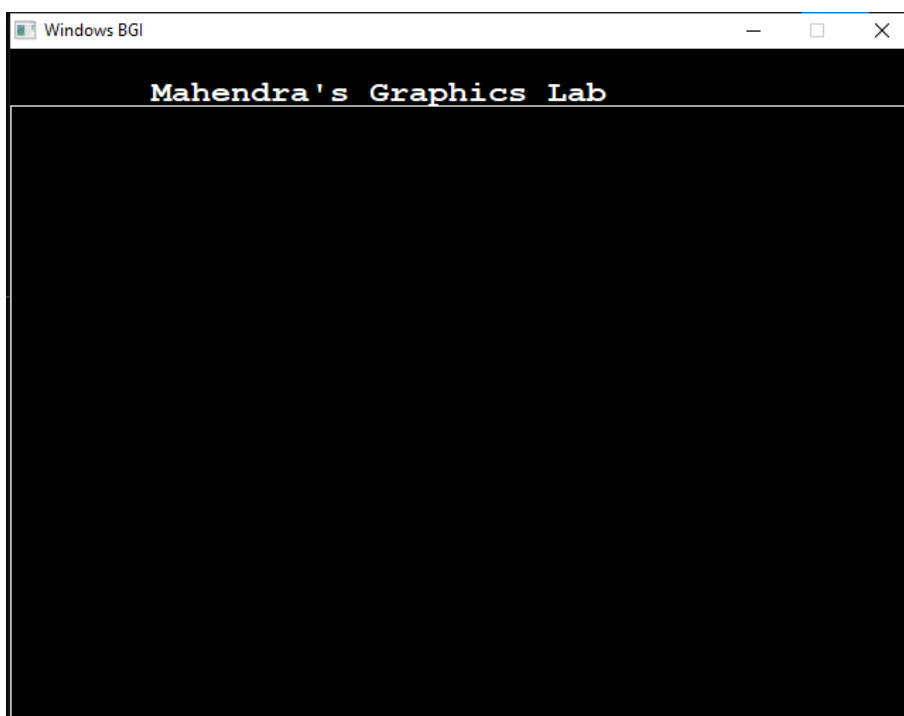
    outtextxy(100, 120, "Press any other key to continue or ESC
to exit.");

    if (choice == 27) {
        break;
    }
}


closegraph();
return 0;
}

```

OUTPUT:



Join This Group



BCA - Tribhuvan University
2 039 members, 31 online

Welcome to the BCA General Help Group!

All Semester Global Group

Join BCA students across Nepal for:

- 💡 Academic Help
- 📖 Study Materials & Notes
- 🚀 Opportunities
- 🎯 Exam Support