WEEK-8

1. #include <stdio.h>

int main() {

int N;

// Read the value of N

scanf("%d", &N);

// Loop to print each line

for (int i = 1; i <= N; i++) {

// Loop to print 1's in each row

for (int j = 1; j <= i; j++) {

printf("1");

}

// Move to the next line after printing each row

printf("\n");

}

return 0;

}

Explanation:

Input: The program first reads an integer N from the user.

Outer Loop: The loop runs i from 1 to N to handle each row.

Inner Loop: The inner loop prints i number of 1s in each row.

New Line: After printing each row, the program moves to the next line.

Example:

Sample Input 1:

Copy

Edit

3

Expected Output 1:

Copy

Edit

1

11

111

Sample Input 2:

Copy

Edit

5

Expected Output 2:

yaml

Copy

Edit

1

11

111

1111

11111

Edge Case:

For N = 0, the program will print nothing.

This updated version meets the new requirement where the number of 1s increases per row.

WEEK-9

1. #include <stdio.h>

int main() {

int n, sum = 0;

scanf("%d", &n);

for(int i = 1; i < n; i++) {

if(n % i == 0) {

sum += i;

}

}

if(sum == n) {

printf("%d is a perfect number\n", n);

} else {

printf("%d is not a perfect number\n", n);

}

return 0;

}

🟢 Output for input 626

626 is not a perfect number#include <stdio.h>

2. int main() {

char str[100], \*p = str;

printf("Enter string: ");

gets(str);

while(\*p) if(\*p >= 'a' && \*p <= 'z') \*p -= 32, p++;

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

return 0;

}

🟢 Sample Input/Output:

Input:

Enter string: welcome

Output:

Transformed string: WELCOME

3. #include <stdio.h>

struct Distance { int feet, inches; };

int main() {

struct Distance d1, d2;

scanf("%d %d %d %d", &d1.feet, &d1.inches, &d2.feet, &d2.inches);

d1.feet += d2.feet + (d1.inches + d2.inches) / 12;

d1.inches = (d1.inches + d2.inches) % 12;

printf("Total distance- Feet: %d, Inches: %d\n", d1.feet, d1.inches);

return 0;

}

🟢 Sample Input/Output:

Input:

23 45 67 89

Output:

Total distance- Feet: 101, Inches: 2

4. #include <stdio.h>

#include <ctype.h>

int main() {

char str[100];

int alphabets = 0, digits = 0, specials = 0;

fgets(str, sizeof(str), stdin);

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

if(isalpha(str[i])) {

alphabets++;

} else if(isdigit(str[i])) {

digits++;

} else if(!isspace(str[i])) {

specials++;

}

}

printf("Alphabets = %d\nNumbers = %d\nSpecial Characters = %d\n", alphabets, digits, specials);

return 0;

}

🟢 Sample Input/Output:

Input:

i love my country @#$% india 123

Output:

Alphabets = 19

Numbers = 3

Special Characters = 4

5. #include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

char str[100];

printf("Enter any string: ");

fgets(str, sizeof(str), stdin);

int start = 0, end = strlen(str) - 1;

while(isspace(str[start])) { start++; }

while(end >= start && isspace(str[end])) { end--; }

str[end + 1] = '\0';

for(int i = 0; i <= end - start; i++) {

str[i] = str[start + i];

}

printf("String after trimming space: |%s|\n", str);

return 0;

}

🟢 Sample Input/Output:

Input:

Enter any string: welcome to c

Output:

String after trimming space: |welcome to c|

(Alternate pgm)

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

char str[100];

int start = 0, end;

printf("Enter any string: ");

fgets(str, sizeof(str), stdin);

end = strlen(str) - 1;

// Removing leading spaces

while(isspace(str[start])) {

start++;

}

// Removing trailing spaces

while(end >= start && isspace(str[end])) {

end--;

}

// Moving characters after trimming

for(int i = 0; i <= end - start; i++) {

str[i] = str[start + i];

}

str[end - start + 1] = '\0'; // Null terminate the string

printf("String before trimming space: |%s|\n", str);

printf("String after trimming space: |%s|\n", str);

return 0;

}

🟢 Sample Input/Output:

Input:

Enter any string: welcome to c

Output:

String before trimming space: | welcome to c |

String after trimming space: |welcome to c|

WEEK-10

1. #include <stdio.h>

int main() {

float balance = 0, amount;

char op;

while(1) {

printf("D->Deposit W->Withdraw B->Balance I->Interest Q->Quit\n");

printf("Enter operation: ");

scanf(" %c", &op);

switch(op) {

case 'D': printf("Enter deposit amount: "); scanf("%f", &amount); balance += amount; break;

case 'W': printf("Enter withdraw amount: "); scanf("%f", &amount); if(amount <= balance) balance -= amount; else printf("Insufficient balance!\n"); break;

case 'B': printf("Balance: Rs.%.2f\n", balance); break;

case 'I': printf("Interest: Rs.%.2f\n", balance \* 0.04); break;

case 'Q': return 0;

default: printf("Invalid operation!\n");

}

}

}

🟢 Sample Input/Output:

Input:

rust

Copy

Edit

D->Deposit W->Withdraw B->Balance I->Interest Q->Quit

Enter operation: D

Enter deposit amount: 10000

D->Deposit W->Withdraw B->Balance I->Interest Q->Quit

Enter operation: B

Balance: Rs.10000.00

D->Deposit W->Withdraw B->Balance I->Interest Q->Quit

Enter operation: I

Interest: Rs.400.00

D->Deposit W->Withdraw B->Balance I->Interest Q->Quit

Enter operation: W

Enter withdraw amount: 3000

D->Deposit W->Withdraw B->Balance I->Interest Q->Quit

Enter operation: B

Balance: Rs.7000.00

D->Deposit W->Withdraw B->Balance I->Interest Q->Quit

Enter operation: Q

2. #include <stdio.h>

int main() {

int r1, c1, r2, c2;

// Input the dimensions of both matrices

scanf("%d%d%d%d", &r1, &c1, &r2, &c2);

// Check if matrix multiplication is possible

if (c1 != r2) {

printf("Matrix multiplication impossible\n");

return 0;

}

int A[r1][c1], B[r2][c2], C[r1][c2];

// Input the elements of the first matrix

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++) {

scanf("%d", &A[i][j]);

}

}

// Input the elements of the second matrix

for (int i = 0; i < r2; i++) {

for (int j = 0; j < c2; j++) {

scanf("%d", &B[i][j]);

}

}

// Matrix multiplication

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

C[i][j] = 0;

for (int k = 0; k < c1; k++) {

C[i][j] += A[i][k] \* B[k][j];

}

}

}

// Output the resultant matrix

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

printf("%d ", C[i][j]);

}

printf("\n");

}

return 0;

}

**🟢 Sample Input/Output:**

**Input:**

2 2 2 2

1 2

3 4

5 6

7 8

**Output:**

19 22

43 50

**Explanation:**

* The program multiplies the two 2x2 matrices:
  + **Matrix A** (2x2):

1 2

3 4

* + **Matrix B** (2x2):

5 6

7 8

* **Matrix Multiplication Result:**
  + First row: (1\*5 + 2\*7) = 19, (1\*6 + 2\*8) = 22
  + Second row: (3\*5 + 4\*7) = 43, (3\*6 + 4\*8) = 50

Thus, the output is:

19 22

43 50

3. #include <stdio.h>

#include <string.h>

struct student {

char name[50];

int regno, marks[5], total;

float percentage;

};

int main() {

struct student s[5];

for (int i = 0; i < 5; i++) {

printf("Enter details for student %d:\n", i+1);

printf("Name: "); getchar(); fgets(s[i].name, sizeof(s[i].name), stdin);

s[i].name[strcspn(s[i].name, "\n")] = 0;

printf("Registration Number: "); scanf("%d", &s[i].regno);

printf("Enter marks for 5 subjects:\n");

s[i].total = 0;

for (int j = 0; j < 5; j++) {

printf("Subject %d: ", j+1);

scanf("%d", &s[i].marks[j]);

s[i].total += s[i].marks[j];

}

s[i].percentage = (float)s[i].total / 5;

}

for (int i = 0; i < 4; i++) {

for (int j = i+1; j < 5; j++) {

if (s[i].total > s[j].total) {

struct student temp = s[i];

s[i] = s[j];

s[j] = temp;

}

}

}

printf("Sorted list of students in ascending order of total marks:\n");

for (int i = 0; i < 5; i++) {

printf("Name: %s, Registration Number: %d, Total Marks: %d, Percentage: %.2f%%\n",

s[i].name, s[i].regno, s[i].total, s[i].percentage);

}

return 0;

}

🟢 Explanation:

Input: The program reads the student’s name, registration number, and marks for 5 subjects.

Processing: It calculates the total marks and percentage for each student.

Sorting: It sorts the students in ascending order based on total marks.

Output: The sorted list is printed with the student's name, registration number, total marks, and percentage.

🟢 Sample Input/Output:

Input:

yaml

Copy

Edit

Enter details for student 1:

Name: SELVAN

Registration Number: 101

Enter marks for 5 subjects:

Subject 1: 90

Subject 2: 98

Subject 3: 98

Subject 4: 87

Subject 5: 86

Enter details for student 2:

Name: KRISH

Registration Number: 302

Enter marks for 5 subjects:

Subject 1: 98

Subject 2: 88

Subject 3: 87

Subject 4: 86

Subject 5: 85

Enter details for student 3:

Name: VINAYKUMAR

Registration Number: 102

Enter marks for 5 subjects:

Subject 1: 76

Subject 2: 68

Subject 3: 66

Subject 4: 88

Subject 5: 77

Enter details for student 4:

Name: RANJITH

Registration Number: 222

Enter marks for 5 subjects:

Subject 1: 55

Subject 2: 56

Subject 3: 57

Subject 4: 54

Subject 5: 50

Enter details for student 5:

Name: KUMAR

Registration Number: 333

Enter marks for 5 subjects:

Subject 1: 35

Subject 2: 56

Subject 3: 67

Subject 4: 33

Subject 5: 52

Output:

yaml

Copy

Edit

Sorted list of students in ascending order of total marks:

Name: KUMAR, Registration Number: 333, Total Marks: 243, Percentage: 48.60%

Name: RANJITH, Registration Number: 222, Total Marks: 272, Percentage: 54.40%

Name: VINAYKUMAR, Registration Number: 102, Total Marks: 375, Percentage: 75.00%

Name: KRISH, Registration Number: 302, Total Marks: 444, Percentage: 88.80%

Name: SELVAN, Registration Number: 101, Total Marks: 459, Percentage: 91.80%

(Alternate pgm)

#include <stdio.h>

#include <string.h>

struct student {

char name[50];

int regno, marks[5], total;

float percentage;

};

int main() {

struct student s[5];

for (int i = 0; i < 5; i++) {

printf("Enter details for student %d:\nName: ", i+1);

getchar(); fgets(s[i].name, sizeof(s[i].name), stdin);

s[i].name[strcspn(s[i].name, "\n")] = 0;

printf("Registration Number: "); scanf("%d", &s[i].regno);

printf("Enter marks for 5 subjects:\n");

s[i].total = 0;

for (int j = 0; j < 5; j++) {

printf("Subject %d: ", j+1); scanf("%d", &s[i].marks[j]);

s[i].total += s[i].marks[j];

}

s[i].percentage = s[i].total / 5.0;

}

for (int i = 0; i < 4; i++) {

for (int j = i+1; j < 5; j++) {

if (s[i].total > s[j].total) {

struct student temp = s[i];

s[i] = s[j];

s[j] = temp;

}

}

}

printf("Sorted students by total marks:\n");

for (int i = 0; i < 5; i++) {

printf("Name: %s, Regno: %d, Total: %d, %.2f%%\n", s[i].name, s[i].regno, s[i].total, s[i].percentage);

}

return 0;

}

🟢 Changes Made to Shorten the Code:

Simplified Input:

getchar() is used to clear the newline character left behind by scanf for the student name input.

No need for separate functions to calculate total and percentage.

In-place Sorting:

The sorting is done directly in the main() function with bubble sort. There’s no separate function call, keeping it more compact.

Efficient Output:

Print statements are reduced to keep the program minimal.

🟢 Sample Input/Output:

Input:

yaml

Copy

Edit

Enter details for student 1:

Name: SELVAN

Registration Number: 101

Enter marks for 5 subjects:

Subject 1: 90

Subject 2: 98

Subject 3: 98

Subject 4: 87

Subject 5: 86

Enter details for student 2:

Name: KRISH

Registration Number: 302

Enter marks for 5 subjects:

Subject 1: 98

Subject 2: 88

Subject 3: 87

Subject 4: 86

Subject 5: 85

Enter details for student 3:

Name: VINAYKUMAR

Registration Number: 102

Enter marks for 5 subjects:

Subject 1: 76

Subject 2: 68

Subject 3: 66

Subject 4: 88

Subject 5: 77

Enter details for student 4:

Name: RANJITH

Registration Number: 222

Enter marks for 5 subjects:

Subject 1: 55

Subject 2: 56

Subject 3: 57

Subject 4: 54

Subject 5: 50

Enter details for student 5:

Name: KUMAR

Registration Number: 333

Enter marks for 5 subjects:

Subject 1: 35

Subject 2: 56

Subject 3: 67

Subject 4: 33

Subject 5: 52

Output:

yaml

Copy

Edit

Sorted students by total marks:

Name: KUMAR, Regno: 333, Total: 243, 48.60%

Name: RANJITH, Regno: 222, Total: 272, 54.40%

Name: VINAYKUMAR, Regno: 102, Total: 375, 75.00%

Name: KRISH, Regno: 302, Total: 444, 88.80%

Name: SELVAN, Regno: 101, Total: 459, 91.80%

4. #include <stdio.h>

int main() {

char op;

float num1, num2;

while(1) { // Start of the loop

printf("Enter an operator (+, -, \*, / ) if want to exit press x: ");

scanf(" %c", &op);

if (op == 'x') { // Check if user wants to exit

break;

}

printf("Enter two operands: ");

scanf("%f %f", &num1, &num2);

switch(op) { // Switch to handle different operations

case '+': {

printf("%.2f\n", num1 + num2);

break;

}

case '-': {

printf("%.2f\n", num1 - num2);

break;

}

case '\*': {

printf("%.2f\n", num1 \* num2);

break;

}

case '/': {

if(num2 != 0) { // Check if division by zero

printf("%.2f\n", num1 / num2);

} else { // Handle division by zero

printf("Error! Division by zero.\n");

}

break;

}

default: {

printf("Error! Please write a valid operator.\n");

break;

}

}

} // End of the loop

return 0;

}

Explanation of the Changes:

Proper Braces for Each Block: Every control structure, including if, else, and switch, is now enclosed with {} to ensure clarity and proper scoping.

Indentation and Code Blocks: Each section of code under the switch-case, if, and else structures is indented appropriately, making it clear which code belongs to which block.

Sample Input/Output:

Input:

pgsql

Copy

Edit

Enter an operator (+, -, \*, / ) if want to exit press x: !

Enter two operands: 12 23

Error! Please write a valid operator.

Enter an operator (+, -, \*, / ) if want to exit press x: +

Enter two operands: 12 23

35.00

Enter an operator (+, -, \*, / ) if want to exit press x: x

Output:

pgsql

Copy

Edit

Error! Please write a valid operator.

35.00

5. #include <stdio.h>

int main() {

int n;

float sum = 0.0;

printf("Input any number: ");

scanf("%d", &n);

for (int i = 1; i <= n; i++) { // Start of the for loop

sum += 1.0 / i; // Add the current term to the sum

if (i == n) { // Check if it's the last term

printf("1/%d\n", i);

} else { // Print terms except the last one

printf("1/%d + ", i);

}

} // End of the for loop

printf("Sum = %.2f\n", sum); // Print the final sum

return 0;

}

Explanation of Changes:

Brackets around loop and condition blocks: I've added {} around all loops and conditional blocks to ensure clarity in scoping.

Indentation: Each block of code inside the loop and if-else statements is indented to maintain readability and ensure proper code structure.

Sample Input/Output:

Input:

pgsql

Copy

Edit

Input any number: 7

Output:

swift

Copy

Edit

1/1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7

Sum = 2.93

6. #include <stdio.h>

int main() {

int n;

printf("Enter the number of elements: ");

scanf("%d", &n);

int arr[n];

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

for (int i = 0; i < n - 1; i++) { // Outer loop

for (int j = i + 1; j < n; j++) { // Inner loop

if (\*(arr + i) > \*(arr + j)) { // If elements are out of order

int temp = \*(arr + i);

\*(arr + i) = \*(arr + j);

\*(arr + j) = temp;

}

}

}

for (int i = 0; i < n; i++) { // Print sorted array

printf("%d ", \*(arr + i));

}

printf("\n");

return 0;

}

Explanation of Changes:

Brackets: Each for loop and if statement now has brackets {} for clarity and better readability.

Pointer Arithmetic: The array elements are accessed using pointer arithmetic (\*(arr + i)).

Sample Input/Output:

Input:

typescript

Copy

Edit

Enter the number of elements: 4

76 54 88 99

Output:

Copy

Edit

54 76 88 99

WEEK-11

1. #include <stdio.h>

void sort(int \*arr, int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = i + 1; j < n; j++) {

if (arr[i] > arr[j]) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

int main() {

int n;

printf("Enter the Number of Elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter the Array elements:\n");

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

sort(arr, n);

printf("After Sorting:\n");

for (int i = 0; i < n; i++) {

printf("%d\n", arr[i]);

}

return 0;

}

Explanation:

Brackets for clarity: The code now has explicit {} brackets around loops and conditionals for better readability and structure.

Pass by reference: The sorting function sort takes a pointer to the array (int \*arr) and modifies the original array directly.

Sample Input/Output:

Input:

mathematica

Copy

Edit

Enter the Number of Elements: 5

Enter the Array elements:

76

66

44

90

1

Output:

yaml

Copy

Edit

After Sorting:

1

44

66

76

90

2. #include <stdio.h>

#include <ctype.h>

int main() {

char str[100];

int v = 0, c = 0;

fgets(str, sizeof(str), stdin);

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

char ch = tolower(str[i]);

if (ch >= 'a' && ch <= 'z') {

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') v++;

else c++;

}

}

printf("Vowels: %d\nConsonants: %d\n", v, c);

return 0;

}

Explanation:

Brackets Added: Brackets are added around the conditional statements for clarity and to ensure proper structure.

Short and Direct: The program is compact, using fgets to read input and a simple loop to count vowels and consonants.

Sample Input/Output:

Input:

csharp

Copy

Edit

Enter the string: india is my country

Output:

makefile

Copy

Edit

Vowels: 6

Consonants: 10

3. #include <stdio.h>

int main() {

FILE \*src = fopen("file1.txt", "r"), \*dest = fopen("file2.txt", "w");

if (!src || !dest) {

return 1;

}

char ch;

while ((ch = fgetc(src)) != EOF) {

fputc(ch, dest);

}

printf("File copied successfully\n");

fclose(src);

fclose(dest);

return 0;

}

Explanation:

Brackets for clarity: I've ensured that brackets are used around the error check and file closing.

Compactness: The program remains concise and efficient while maintaining clarity with the use of brackets.

Sample Input/Output:

Input (file1.txt):

css

Copy

Edit

INDIA IS MY COUNTRY. I LOVE MY COUNTRY

Output:

arduino

Copy

Edit

File copied successfully

4. #include <stdio.h>

#include <stdlib.h>

int main() {

int n;

float \*arr, largest;

scanf("%d", &n);

arr = (float \*)malloc(n \* sizeof(float));

for (int i = 0; i < n; i++) {

scanf("%f", &arr[i]);

}

largest = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] > largest) {

largest = arr[i];

}

}

printf("Largest number = %.2f\n", largest);

free(arr);

return 0;

}

Explanation:

Brackets around all code blocks: Added brackets to enhance clarity and follow standard coding practices.

Compact code: The program is minimal while keeping the essential functionality, with clear structure and brackets.

Sample Input/Output:

Input:

Copy

Edit

5

3.4 2.4 -5 24.2 6.7

Output:

java

Copy

Edit

Largest number = 24.20

5. #include <graphics.h>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

int main() {

int gd = DETECT, gm;

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

srand(time(0));

while (!kbhit()) {

int x = rand() % getmaxx();

int y = rand() % getmaxy();

setcolor(WHITE);

circle(x, y, 1); // Draw a small star (circle)

delay(200); // Twinkle speed

cleardevice(); // Clear screen for animation

}

closegraph();

return 0;

}

Explanation:

Twinkling Stars: Random stars are created by drawing small circles (radius = 1) at random positions.

Screen Clears: The screen is cleared (cleardevice()) after every star is drawn to make it look like twinkling.

Exit on Keypress: The loop continues until a key is pressed (kbhit()).

Simplifications:

Removed unnecessary variables: No extra code, just random positions and simple star drawing.

Star size: The stars are now fixed to a radius of 1 for simplicity.

No functions: Everything is done in the main() loop.

Sample Output:

Stars will randomly twinkle on the screen and disappear immediately, creating a simple animation. The program exits when you press any key.

This is about as simple as it gets while still

(Alternate pgm)

To create a C program that draws animated twinkling stars using the graphics.h library, we need to first ensure that you have the **graphics library** properly set up. This library is not part of the standard C library, and it is mainly used in Turbo C or old IDEs that support it.

### Steps to Set Up graphics.h:

1. **Install Turbo C++ or Code::Blocks** (with MinGW) with support for graphics.h.
2. **Link the graphics.h library** in your IDE settings.

Here's a simple example program that creates an **animated twinkling stars** effect:

### ****C Program to Draw Animated Twinkling Stars****

c

CopyEdit

#include <graphics.h>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

void drawStar(int x, int y, int size) {

setcolor(WHITE);

for (int i = 0; i < size; i++) {

circle(x, y, i); // Draw circles of increasing size to create a star-like effect

}

}

int main() {

int gd = DETECT, gm;

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

int x, y, size;

srand(time(0));

while (!kbhit()) {

x = rand() % getmaxx();

y = rand() % getmaxy();

size = rand() % 3 + 1; // Random star size between 1 and 3

drawStar(x, y, size);

delay(200); // Control the speed of twinkling

// Clear the screen to animate the twinkling effect

cleardevice();

}

closegraph();

return 0;

}

### ****Explanation:****

* **Graphics Initialization:** The program uses initgraph to initialize the graphics mode.
* **Random Star Positions and Sizes:** The program randomly places stars on the screen and adjusts their size using random values.
* **Twinkling Effect:** The stars are drawn in different locations, and the screen is cleared every 200 milliseconds to create the twinkling effect.
* **User Exit:** The program exits when a key is pressed (detected by kbhit()).

### ****Libraries:****

* **graphics.h**: To draw on the screen.
* **conio.h**: For kbhit() to detect keyboard input and control the exit.
* **stdlib.h**: For rand() to create random numbers.
* **time.h**: For random number seeding using time(0) to ensure different results on each run.

### ****Sample Output:****

The screen will display stars that appear at random positions, twinkling in and out.

### ****Important Notes:****

* **Platform Compatibility:** The graphics.h library is outdated and primarily works on old compilers like Turbo C++. Modern compilers (like GCC or Clang) do not support graphics.h natively.
* **Alternative:** If you're using a more modern compiler (like GCC with Code::Blocks or Visual Studio), you may need to use libraries like **SDL**, **SFML**, or **OpenGL** to handle graphics.

This program uses **simple animation** techniques to simulate twinkling stars in a C program using graphics.h.

6. #include <stdio.h>

int main() {

int n1, n2;

scanf("%d", &n1);

int arr1[n1];

for (int i = 0; i < n1; i++) {

scanf("%d", &arr1[i]);

}

scanf("%d", &n2);

int arr2[n2];

for (int i = 0; i < n2; i++) {

scanf("%d", &arr2[i]);

}

int merged[n1 + n2];

for (int i = 0; i < n1; i++) {

merged[i] = arr1[i];

}

for (int i = 0; i < n2; i++) {

merged[n1 + i] = arr2[i];

}

for (int i = 0; i < n1 + n2 - 1; i++) {

for (int j = i + 1; j < n1 + n2; j++) {

if (merged[i] > merged[j]) {

int temp = merged[i];

merged[i] = merged[j];

merged[j] = temp;

}

}

}

for (int i = 0; i < n1 + n2; i++) {

printf("%d ", merged[i]);

}

return 0;

}

Explanation:

Input: The program takes two arrays and reads their values.

Merge: It then merges the two arrays into a new merged[] array.

Sorting: The merged array is sorted using bubble sort with proper brackets.

Output: Finally, the sorted merged[] array is printed.

Sample Input:

Copy

Edit

5

66 44 88 98 12

4

87 65 43 21

Sample Output:

Copy

Edit

12 21 43 44 65 66 87 88 98