

K PRASANTH (192211920) CSA0699 DAA

1.FIBONACCI SERIES USING RECURSION

```
#include <stdio.h>

int fibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n, i;
    printf("Enter the number of terms: ");
    scanf("%d", &n);
    printf("Fibonacci series: ");
    for (i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program named 'fibo using recursion.cpp'. The code defines a recursive function 'fibonacci' and a main function that prints the first 'n' terms of the Fibonacci series. On the right, the terminal window shows the execution results: 'Fibonacci Series: 0, 1, 1, 2, 3,' followed by a message about the process exiting and a prompt to press any key. Below the terminal is a 'Compiler' tab showing compilation results with 0 errors and 0 warnings, and a file size of 128.6572265625 KiB. The taskbar at the bottom shows various application icons.

```

1 #include <stdio.h>
2 int fibonacci(int n) {
3     if (n <= 1) {
4         return n;
5     }
6     return fibonacci(n - 1) + fibonacci(n - 2);
7 }
8 int main() {
9     int n, i;
10    printf("Enter the number of terms: ");
11    scanf("%d", &n);
12    printf("Fibonacci series: ");
13    for (i = 0; i < n; i++) {
14        printf("%d ", fibonacci(i));
15    }
16    return 0;
17 }
18

```

2. ARMSTRONG NUMBER

```
#include<stdio.h>

int main()

{
    int num,originalNum,remainder,result=0;
    scanf("%d",&num);
    originalNum=num;

    while(originalNum!=0)

    {
        remainder=originalNum%10;
        result+=remainder*remainder*remainder;
        originalNum/=10;
    }
}
```

```

if(result==num)

    printf("%d is an Armstrong number.",num);

else

    printf("%d is not an Armstrong number.",num);

return 0;

}

```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C program named 'Untitled1' with code to check if a number is Armstrong. The code includes a while loop to calculate digits and their powers, followed by an if-else block to print the result. The status bar at the bottom shows compilation results: 0 errors, 0 warnings, output filename 'C:\Users\khato\OneDrive\Documents\os\armstrong.exe', output size '128.1015625 Kib', and compilation time '0.22s'. A terminal window shows the program's output for the input '153', which is identified as an Armstrong number. The title bar indicates the file is executing.

```

1 #include<stdio.h>
2 int main()
3 {
4     int num,originalNum,remainder,result=0;
5     scanf("%d",&num);
6     originalNum=num;
7
8     while(originalNum!=0)
9     {
10         remainder=originalNum%10;
11         result+=remainder*remainder*remainder;
12         originalNum/=10;
13     }
14     if(result==num)
15         printf("%d is an Armstrong number.",num);
16     else
17         printf("%d is not an Armstrong number.",num);
18     return 0;
19 }

```

3. GCD OF TWO NUMBERS

```

#include <stdio.h>

int main()

{

int a,b;

printf("enter the two numbers:");

scanf("%d %d",&a,&b);

while(b){


```

```

int temp=b;
b=a%b;
a=temp;
}

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

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C++ program for calculating the Greatest Common Divisor (GCD) using the Euclidean algorithm. The code includes comments, variable declarations, and a while loop for the iterative process. On the right, a terminal window shows the execution of the program. It prompts the user to enter two numbers (2 and 4), performs the calculation, and outputs the result (gcd is 2). Below the terminal is a compilation log window showing successful compilation with no errors or warnings.

```

C:\Users\khato\OneDrive\Documents\os\gcd.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp
C:\Users\khato\OneDrive\Documents\os\gcd.cpp
1 #include <stdio.h>
2 int main()
3 {
4     int a,b;
5     printf("enter the two numbers:");
6     scanf("%d %d",&a,&b);
7     while(b!=0)
8     {
9         int temp=b;
10        b=a%b;
11        a=temp;
12    }
13    printf("gcd is %d\n",a);
14    return 0;
}

enter the two numbers: 2 4
gcd is 2
-----
Process exited after 3.223 seconds with return
value 0
Press any key to continue . . .

```

```

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\gcd.exe
- Output Size: 128.1015625 Kib
- Compilation Time: 0.27s

```

4. LARGEST ELEMENT IN AN ARRAY

```

#include <stdio.h>

int main() {

    int n, i;

    float arr[100];

    printf("Enter the number of elements (up to 100): ");

    scanf("%d", &n);

```

```

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

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

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

float max = arr[0];

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

    if (arr[i] > max) {

        max = arr[i];
    }
}

printf("The largest element is: %.2f\n", max);

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\khato\OneDrive\Documents\os\largest element.cpp - [Executing] - Dev-C++ 5.11
- Toolbar:** Standard Dev-C++ toolbar with icons for file operations, project management, and compilation.
- MenuBar:** File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help.
- StatusBar:** Line: 11, Col: 24, Sek: 0, Lines: 21, Length: 462, Insert, Done parsing in 0 seconds.
- Code Editor:** Untitled1 - perfect number using if else.cpp (highlighted). The code is identical to the one provided above.
- Output Window:** Shows the execution of the program. It prompts for the number of elements (6), then the elements (7, 8, 9, 15, 17, 18), and finally outputs "The largest element is: 18.00".
- Compiler Log:** Shows compilation results with 0 errors and 0 warnings, output filename C:\Users\khato\OneDrive\Documents\os\largest element.exe, output size 128.7900390625 KiB, and compilation time 0.25s.

5. FACTORIAL

```
#include<stdio.h>

int main()
{
    int num,fact=1;

    printf("enter number:");
    scanf("%d",&num);

    for(int i=1;i<=num;i++){
        fact=fact*i;
    }

    printf("%d\n",fact);

    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The code editor window displays the factorial.cpp file with the provided C code. To the right, a terminal window titled 'C:\Users\khato\OneDrive\Documents\os\factorial.cpp' shows the program's output: 'enter number: 120' followed by the result 'Process exited after 1.201 seconds with return value 0'. Below the terminal is a 'Compiler' tab in the status bar, which shows 'Compilation results...' and details like 'Errors: 0', 'Output Filename: C:\Users\khato\OneDrive\Documents\os\factorial.exe', and 'Compilation Time: 0.27s'. The status bar also includes line, column, and search information.

6. PRIME NUMBER

```
#include<stdio.h>

int main(){
    int n,i,flag=0;
    printf("enter a number");
    scanf("%d",&n);
    if(n==0 || n==1)
        flag=1;
    for(i=2;i<=n/2;++i){
        if(n%i==0){
            flag=1;
            break;
        }
    }
    if(flag==0)
        printf("%d is a prime",n);
    else
        printf("%d is a non prime",n);
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C program named 'prime.cpp' with code to determine if a number is prime. The code includes input handling, a loop for checking divisors, and output statements for prime or non-prime status. Below the editor is a terminal window showing the program's execution and output for the number 3. The bottom panel shows the compiler results, indicating no errors or warnings.

```

1 #include<stdio.h>
2 int main(){
3     int n,i,flag=0;
4     printf("enter a number");
5     scanf("%d",&n);
6     if(n==0||n==1)
7         flag=1;
8     for(i=2;i<=n/2;i++){
9         if(n%i==0){
10             flag=1;
11             break;
12         }
13     }
14     if(flag==0)
15         printf("%d is a prime",n);
16     else
17         printf("%d is a non prime",n);
18     return 0;
19 }

```

enter a number3
3 is a prime
Process exited after 1.328 seconds with return value 0
Press any key to continue . . .

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\prime
- Output Size: 128.1015625 Kib
- Compilation Time: 0.27s

7. SELECTION SORT

```
#include <stdio.h>

void swap(int *a, int *b) {

    int temp = *a;

    *a = *b;

    *b = temp;

}

void selectionSort(int arr[], int n) {

    int i, j, minIndex;

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

        minIndex = i;

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

            if (arr[j] < arr[minIndex]) {

                minIndex = j;
            }
        }
    }

    int temp = arr[i];
    arr[i] = arr[minIndex];
    arr[minIndex] = temp;
}
```

```
    }

}

swap(&arr[minIndex], &arr[i]);

}

void printArray(int arr[], int size) {

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

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

    }

    printf("\n");

}

int main() {

    int arr[] = {64, 25, 12, 22, 11};

    int n = sizeof(arr)/sizeof(arr[0]);



    printf("Original array: \n");

    printArray(arr, n);



    selectionSort(arr, n);



    printf("Sorted array: \n");

    printArray(arr, n);

}
```

```
return 0;  
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ code editor with the following code:

```
if (arr[j] < arr[minIndex]) {  
    minIndex = j;  
}  
swap(&arr[minIndex], &arr[i]);  
}  
void printArray(int arr[], int size) {  
    for (int i = 0; i < size; i++) {  
        printf("%d ", arr[i]);  
    }  
    printf("\n");  
}  
int main() {  
    int arr[] = {64, 25, 12, 22, 11};  
    int n = sizeof(arr)/sizeof(arr[0]);  
    printf("Original array: \n");  
    printArray(arr, n);  
    selectionSort(arr, n);  
    printf("Sorted array: \n");  
    printArray(arr, n);  
    return 0;  
}
```

To the right of the code editor is a terminal window showing the program's output:

```
Original array:  
64 25 12 22 11  
Sorted array:  
11 12 22 25 64  
-----  
Process exited after 0.05808 seconds with return value 0  
Press any key to continue . . .
```

Below the code editor is a status bar with the following information:

```
Line: 15 Col: 10 Sek: 0 Lines: 40 Length: 819 Insert Done parsing in 0.016 seconds
```

At the bottom left is a compilation results window:

```
Compilation results...  
-----  
- Errors: 0  
- Warnings: 0  
- Output Filenam: C:\Users\khato\OneDrive\Documents\os\selection sort.exe  
- Output Size: 129.875 Kib  
- Compilation Time: 0.25s
```

8. BUBBLE SORT

```
#include <stdio.h>  
  
void bubbleSort(int arr[], int n) {  
  
    int i, j, temp;  
  
    for (i = 0; i < n-1; i++) {  
  
        for (j = 0; j < n-i-1; j++) {  
  
            if (arr[j] > arr[j+1]) {  
  
                temp = arr[j];  
  
                arr[j] = arr[j+1];  
  
                arr[j+1] = temp;  
            }  
        }  
    }  
}
```

```
    }

}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);

    printf("Original array: \n");
    printArray(arr, n);

    bubbleSort(arr, n);

    printf("Sorted array: \n");
    printArray(arr, n);

    return 0;
}
```

```

C:\Users\khato\OneDrive\Documents\os\bubble sort.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp
7     temp = arr[j];
8     arr[j] = arr[j+1];
9     arr[j+1] = temp;
10    }
11   }
12 }
13 }
14 void printArray(int arr[], int n) {
15 for (int i = 0; i < n; i++) {
16     printf("%d ", arr[i]);
17 }
18 printf("\n");
19 }
20
21 int main() {
22     int arr[] = {64, 34, 25, 12, 22, 11, 90};
23     int n = sizeof(arr)/sizeof(arr[0]);
24
25     printf("Original array: \n");
26     printArray(arr, n);
27
28     bubbleSort(arr, n);
29
30     printf("Sorted array: \n");
31     printArray(arr, n);
32
33     return 0;
34 }

```

Original array:
64 34 25 12 22 11 90
Sorted array:
11 12 22 25 34 64 90

Process exited after 0.05778 seconds with return value 0
Press any key to continue . . .

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\bubble sort.exe
- Output Size: 129.8427734375 KiB
- Compilation Time: 0.30s

Line: 26 Cob: 24 Sek: 0 Lines: 35 Length: 734 Insert Done parsing in 0 seconds

9. MATRIX MULTIPLICATION

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

```
#define MAX 10 // Maximum size of the matrix
```

```

void matrixMultiplication(int firstMatrix[MAX][MAX], int
secondMatrix[MAX][MAX], int result[MAX][MAX], int row1, int col1, int row2, int
col2) {

// Initialize the result matrix to zero

for (int i = 0; i < row1; i++) {

    for (int j = 0; j < col2; j++) {

        result[i][j] = 0;
    }
}
}
```

```
// Multiply the two matrices

for (int i = 0; i < row1; i++) {
    for (int j = 0; j < col2; j++) {
        for (int k = 0; k < col1; k++) {
            result[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
        }
    }
}

int main() {
    int firstMatrix[MAX][MAX], secondMatrix[MAX][MAX], result[MAX][MAX];
    int row1, col1, row2, col2;

    // Input dimensions of first matrix
    printf("Enter rows and columns of first matrix: ");
    scanf("%d %d", &row1, &col1);

    // Input elements of first matrix
    printf("Enter elements of first matrix:\n");
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col1; j++) {
            scanf("%d", &firstMatrix[i][j]);
        }
    }
}
```

```
}

// Input dimensions of second matrix
printf("Enter rows and columns of second matrix: ");
scanf("%d %d", &row2, &col2);

// Matrix multiplication condition: col1 should be equal to row2
if (col1 != row2) {
    printf("Error! Matrix multiplication is not possible.\n");
    return -1;
}

// Input elements of second matrix
printf("Enter elements of second matrix:\n");
for (int i = 0; i < row2; i++) {
    for (int j = 0; j < col2; j++) {
        scanf("%d", &secondMatrix[i][j]);
    }
}

// Perform matrix multiplication
matrixMultiplication(firstMatrix, secondMatrix, result, row1, col1, row2, col2);

// Output the result matrix
```

```

printf("Resultant matrix:\n");

for (int i = 0; i < row1; i++) {

    for (int j = 0; j < col2; j++) {

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

    printf("\n");
}

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\khato\OneDrive\Documents\os\MATRIX MUL.cpp - [Executing] - Dev-C++ 5.11
- Menu Bar:** File Edit Search View Project Execute Tools AStyle Window Help
- Toolbar:** Standard Dev-C++ toolbar.
- Project Bar:** Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp largest element.cpp factorial.cpp prime.cpp selection sort.cpp bubble sort.cpp MATRIX MUL.cpp
- Code Editor:** Displays the C++ code for matrix multiplication.
- Output Window:** Shows the command-line interface interaction and the resulting matrix output.
- Compiler Log:** Shows compilation results with 0 errors and 0 warnings.
- System Taskbar:** Shows weather (94°F Mostly cloudy), system icons, and system status (ENG IN, 1:58 PM, 9/19/2024).

```

#include <stdio.h>

#define MAX 10 // Maximum size of the matrix

void matrixMultiplication(int firstMatrix[MAX][MAX], int secondMatrix[MAX][MAX]) {
    // Initialize the result matrix to zero
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col2; j++) {
            result[i][j] = 0;
        }
    }

    // Multiply the two matrices
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col2; j++) {
            for (int k = 0; k < col1; k++) {
                result[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
            }
        }
    }
}

int main() {
    int firstMatrix[MAX][MAX], secondMatrix[MAX][MAX], result[MAX][MAX];
    int row1, col1, row2, col2;

    // Input dimensions of first matrix
    printf("Enter rows and columns of first matrix: ");
    scanf("%d %d", &row1, &col1);

    // Input dimensions of second matrix
    printf("Enter rows and columns of second matrix: ");
    scanf("%d %d", &row2, &col2);

    // Input elements of first matrix
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col1; j++) {
            printf("Enter element (%d,%d): ", i+1, j+1);
            scanf("%d", &firstMatrix[i][j]);
        }
    }

    // Input elements of second matrix
    for (int i = 0; i < row2; i++) {
        for (int j = 0; j < col2; j++) {
            printf("Enter element (%d,%d): ", i+1, j+1);
            scanf("%d", &secondMatrix[i][j]);
        }
    }

    // Perform matrix multiplication
    matrixMultiplication(firstMatrix, secondMatrix);

    // Output resultant matrix
    printf("Resultant matrix:\n");
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col2; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
}

Process exited after 10.49 seconds with return value 0
Press any key to continue . . .

```

PALLINDROME

```

#include <stdio.h>

#include <string.h>

#include <ctype.h>

```

```
void checkPalindrome(char str[]) {  
    int len = strlen(str);  
    int start = 0;  
    int end = len - 1;  
    int isPalindrome = 1; // Assume the string is a palindrome  
  
    while (start < end) {  
        // Ignore non-alphanumeric characters and make it case-insensitive  
        while (start < end && !isalnum(str[start])) start++;  
        while (start < end && !isalnum(str[end])) end--;  
  
        if (tolower(str[start]) != tolower(str[end])) {  
            isPalindrome = 0; // Not a palindrome  
            break;  
        }  
        start++;  
        end--;  
    }  
  
    if (isPalindrome) {  
        printf("The string is a palindrome.\n");  
    } else {  
        printf("The string is not a palindrome.\n");  
    }  
}
```

```

    }

}

int main() {
    char str[100];

    printf("Enter a string: ");

    fgets(str, sizeof(str), stdin); // Use fgets to allow spaces in the input
    str[strcspn(str, "\n")] = '\0'; // Remove newline character if present

    checkPalindrome(str);

    return 0;
}

```

The screenshot shows the Dev-C++ IDE interface with two windows. The left window is the code editor displaying the C program for checking palindromes. The right window is a terminal window showing the execution of the program.

Code Editor (Left Window):

```

14     while (start < end && !isalnum(str[end])) end--;
15
16     if (tolower(str[start]) != tolower(str[end])) {
17         isPalindrome = 0; // Not a palindrome
18         break;
19     }
20     start++;
21     end--;
22 }
23
24 if (isPalindrome) {
25     printf("The string is a palindrome.\n");
26 } else {
27     printf("The string is not a palindrome.\n");
28 }
29
30
31 int main() {
32     char str[100];
33
34     printf("Enter a string: ");
35     fgets(str, sizeof(str), stdin); // Use fgets to allow
36     str[strcspn(str, "\n")] = '\0'; // Remove newline character
37
38     checkPalindrome(str);
39
40     return 0;
41 }

```

Terminal Window (Right Window):

```

C:\Users\khato\OneDrive\Documents\os\PALLINDROME.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug
C:\Users\khato\OneDrive\Doc x + v
Enter a string: MADAM
The string is a palindrome.

Process exited after 3.793 seconds with return value 0
Press any key to continue . . .

```

The terminal window shows the output of the program when it is run with the input "MADAM". The program correctly identifies it as a palindrome.

COPY THE STRING

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

int main() {
    char source[100], destination[100];

    // Input string from user
    printf("Enter a string: ");
    fgets(source, sizeof(source), stdin);

    // Remove the trailing newline character that fgets adds
    source[strcspn(source, "\n")] = 0;

    // Copy the source string to destination
    strcpy(destination, source);

    // Output the copied string
    printf("Copied string: %s\n", destination);

    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program for copying strings. On the right, a terminal window shows the output of the program. Below the terminal is a 'Compiler' tab showing compilation results.

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int main() {
5     char source[100], destination[100];
6
7     // Input string from user
8     printf("Enter a string: ");
9     fgets(source, sizeof(source), stdin);
10
11    // Remove the trailing newline character that fgets adds
12    source[strcspn(source, "\n")] = 0;
13
14    // Copy the source string to destination
15    strcpy(destination, source);
16
17    // Output the copied string
18    printf("Copied string: %s\n", destination);
19
20    return 0;
21 }

```

Terminal Output:

```

Enter a string: PRASANTH
Copied string: PRASANTH
-----
Process exited after 4.506 seconds with return
value 0
Press any key to continue . . .

```

Compiler Tab:

```

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\COPY STRING.exe
- Output Size: 128.9443359375 Kib
- Compilation Time: 0.52s

```

BINARY SEARCH

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

```
// Function to perform binary search
```

```
int binarySearch(int arr[], int size, int target) {
```

```
    int left = 0;
```

```
    int right = size - 1;
```

```
    while (left <= right) {
```

```
        int mid = left + (right - left) / 2; // Find the middle index
```

```
        // Check if target is present at mid
```

```
        if (arr[mid] == target)
```

```
            return mid; // Target found
```

```
// If target is greater, ignore the left half
if (arr[mid] < target)

    left = mid + 1;

// If target is smaller, ignore the right half
else

    right = mid - 1;

}

return -1; // Target not found
```

```
int main() {

    int arr[] = {2, 3, 4, 10, 40};

    int size = sizeof(arr) / sizeof(arr[0]);

    int target = 10;

    // Perform binary search

    int result = binarySearch(arr, size, target);

    // Output result

    if (result != -1)

        printf("Element found at index %d\n", result);
```

```

else
printf("Element not found in the array\n");

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C++ file named `BINARY SEARCH.cpp`. The code implements a binary search algorithm. The output window shows the program's execution results: "Element found at index 3". The compiler log shows no errors or warnings.

```

C:\Users\khato\OneDrive\Documents\os\BINARY SEARCH.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASyntax Window Help
C:\Users\khato\OneDrive\Documents\os\globals
Project Classes Debug Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp largest element.cpp factorial.cpp prime.cpp selection sort.cpp bubble sort.cpp MATRIX MUL.cpp COPY STRING.cpp
C:\Users\khato\OneDrive\Doc
Element found at index 3
-----
Process exited after 0.05328 seconds with return value
0
Press any key to continue . . .

```

```

16     if (arr[mid] < target)
17         left = mid + 1;
18
19     // If target is smaller, ignore the right half
20     else
21         right = mid - 1;
22     }
23
24     return -1; // Target not found
25 }
26
27 int main() {
28     int arr[] = {2, 3, 4, 10, 40};
29     int size = sizeof(arr) / sizeof(arr[0]);
30     int target = 10;
31
32     // Perform binary search
33     int result = binarySearch(arr, size, target);
34
35     // Output result
36     if (result != -1)
37         printf("Element found at index %d\n", result);
38     else
39         printf("Element not found in the array\n");
40
41     return 0;
42 }

```

REVERSE STRING

```

#include <stdio.h>

#include <string.h>

void reverseString(char str[]) {

    int n = strlen(str);

    for (int i = 0; i < n / 2; i++) {

        // Swap characters

        char temp = str[i];

```

```
    str[i] = str[n - i - 1];
    str[n - i - 1] = temp;
}

int main() {
    char str[100];

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin); // Read input string
    str[strcspn(str, "\n")] = 0; // Remove newline character if present

    reverseString(str); // Call function to reverse the string

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

    return 0;
}
```

```

C:\Users\khato\OneDrive\Documents\os\REVERSE STRING.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp largest element.cpp factorial.cpp prime.cpp selection sort.cpp bubble sort.cpp MATRIX MUL.cpp REVERSE STRING.cpp
PALLINDROME.cpp COPY STRING.cpp BINARY SEARCH.cpp
1 #include <stdio.h>
2 #include <string.h>
3
4 void reverseString(char str[]) {
5     int n = strlen(str);
6     for (int i = 0; i < n / 2; i++) {
7         // Swap characters
8         char temp = str[i];
9         str[i] = str[n - i - 1];
10        str[n - i - 1] = temp;
11    }
12}
13
14 int main() {
15     char str[100];
16
17     printf("Enter a string: ");
18     fgets(str, sizeof(str), stdin); // Read input string
19     str[strcspn(str, "\n")] = 0; // Remove newline character if present
20
21     reverseString(str); // Call function to reverse the string
22
23     printf("Reversed string: %s\n", str);
24
25     return 0;
26 }

```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\REVERSE STRING.exe
- Output Size: 129.310546875 KiB
- Compilation Time: 0.28s

LENGTH OF STRING

```
#include <stdio.h>

#include <string.h> // For strlen
```

```
// Function to find length of string manually

int stringLength(char str[]) {

    int length = 0;

    while (str[length] != '\0') { // '\0' is the null character indicating the end of the
        string

        length++;
    }

    return length;
}
```

```
int main() {  
    char str[100]; // Declaring a string of maximum size 100  
  
    // Input string from user  
    printf("Enter a string: ");  
    fgets(str, sizeof(str), stdin); // Using fgets to safely get input  
  
    // Remove newline character from fgets  
    str[strcspn(str, "\n")] = '\0';  
  
    // Using built-in function strlen  
    printf("Length of the string using strlen: %lu\n", strlen(str));  
  
    // Using manual method  
    printf("Length of the string manually: %d\n", stringLength(str));  
  
    return 0;  
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ source code file named 'LENGTH OF STRING.cpp'. The code implements three methods to calculate the length of a string: a manual character-by-character loop, the built-in `strlen` function, and a manual method using `strcspn`. The code also includes a `main` function that reads a string from the user and prints its length using all three methods. A terminal window shows the output for the string 'PRASANTH', where the lengths are 8, 8, and 8 respectively. Below the terminal is a 'Compiler' tab showing compilation results with 0 errors and 0 warnings, and an output size of 129.830078125 KiB.

```

C:\Users\khato\OneDrive\Documents\os\LENGTH OF STRING.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug
Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp
PALLINDROME.cpp COPY STRING.cpp
4 // Function to find length of string manually
5 int stringLength(char str[]) {
6     int length = 0;
7     while (str[length] != '\0') { // '\0' is the null character
8         length++;
9     }
10    return length;
11 }
12
13 int main() {
14     char str[100]; // Declaring a string of maximum size 100
15
16     // Input string from user
17     printf("Enter a string: ");
18     fgets(str, sizeof(str), stdin); // Using fgets to safely get the string
19
20     // Remove newline character from fgets
21     str[strcspn(str, "\n")] = '\0';
22
23     // Using built-in function strlen
24     printf("Length of the string using strlen: %lu\n", strlen(str));
25
26     // Using manual method
27     printf("Length of the string manually: %d\n", stringLength(str));
28
29 }
30
31

```

C:\Users\khato\OneDr X + - □ X

Enter a string: PRASANTH
Length of the string using strlen: 8
Length of the string manually: 8

Process exited after 4.389 seconds with return value 0
Press any key to continue . . . |

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\LENGTH OF STRING.exe
- Output Size: 129.830078125 KiB
- Compilation Time: 0.23s

Line: 31 Col: 1 Sek: 0 Lines: 31 Length: 841 Insert Done parsing in 0.016 seconds

STRASSEN MATRIX

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

```
#include <stdlib.h>
```

```
// Function to add two matrices
```

```
void addMatrix(int n, int A[][n], int B[][n], int result[][n]) {
```

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

```
        for (int j = 0; j < n; j++)
```

```
            result[i][j] = A[i][j] + B[i][j];
```

```
}
```

```
// Function to subtract two matrices
```

```
void subtractMatrix(int n, int A[][n], int B[][n], int result[][n]) {
```

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

```

for (int j = 0; j < n; j++) {
    result[i][j] = A[i][j] - B[i][j];
}

// Strassen's matrix multiplication function
void strassenMultiply(int n, int A[][n], int B[][n], int C[][n]) {
    // Base case: 2x2 matrix multiplication
    if (n == 2) {
        int P1 = (A[0][0] + A[1][1]) * (B[0][0] + B[1][1]);
        int P2 = (A[1][0] + A[1][1]) * B[0][0];
        int P3 = A[0][0] * (B[0][1] - B[1][1]);
        int P4 = A[1][1] * (B[1][0] - B[0][0]);
        int P5 = (A[0][0] + A[0][1]) * B[1][1];
        int P6 = (A[1][0] - A[0][0]) * (B[0][0] + B[0][1]);
        int P7 = (A[0][1] - A[1][1]) * (B[1][0] + B[1][1]);

        C[0][0] = P1 + P4 - P5 + P7;
        C[0][1] = P3 + P5;
        C[1][0] = P2 + P4;
        C[1][1] = P1 + P3 - P2 + P6;
        return;
    }
}

int newSize = n / 2;

```

```

int A11[newSize][newSize], A12[newSize][newSize], A21[newSize][newSize],
A22[newSize][newSize];

int B11[newSize][newSize], B12[newSize][newSize], B21[newSize][newSize],
B22[newSize][newSize];

int C11[newSize][newSize], C12[newSize][newSize], C21[newSize][newSize],
C22[newSize][newSize];

int P1[newSize][newSize], P2[newSize][newSize], P3[newSize][newSize],
P4[newSize][newSize], P5[newSize][newSize], P6[newSize][newSize],
P7[newSize][newSize];

int temp1[newSize][newSize], temp2[newSize][newSize];

// Divide A and B matrices into 4 sub-matrices
for (int i = 0; i < newSize; i++) {
    for (int j = 0; j < newSize; j++) {
        A11[i][j] = A[i][j];
        A12[i][j] = A[i][j + newSize];
        A21[i][j] = A[i + newSize][j];
        A22[i][j] = A[i + newSize][j + newSize];

        B11[i][j] = B[i][j];
        B12[i][j] = B[i][j + newSize];
        B21[i][j] = B[i + newSize][j];
        B22[i][j] = B[i + newSize][j + newSize];
    }
}

```

```
// P1 = (A11 + A22) * (B11 + B22)
addMatrix(newSize, A11, A22, temp1);
addMatrix(newSize, B11, B22, temp2);
strassenMultiply(newSize, temp1, temp2, P1);

// P2 = (A21 + A22) * B11
addMatrix(newSize, A21, A22, temp1);
strassenMultiply(newSize, temp1, B11, P2);

// P3 = A11 * (B12 - B22)
subtractMatrix(newSize, B12, B22, temp1);
strassenMultiply(newSize, A11, temp1, P3);

// P4 = A22 * (B21 - B11)
subtractMatrix(newSize, B21, B11, temp1);
strassenMultiply(newSize, A22, temp1, P4);

// P5 = (A11 + A12) * B22
addMatrix(newSize, A11, A12, temp1);
strassenMultiply(newSize, temp1, B22, P5);

// P6 = (A21 - A11) * (B11 + B12)
subtractMatrix(newSize, A21, A11, temp1);
addMatrix(newSize, B11, B12, temp2);
```

```
strassenMultiply(newSize, temp1, temp2, P6);
```

```
// P7 = (A12 - A22) * (B21 + B22)  
subtractMatrix(newSize, A12, A22, temp1);  
addMatrix(newSize, B21, B22, temp2);  
strassenMultiply(newSize, temp1, temp2, P7);
```

```
// C11 = P1 + P4 - P5 + P7  
addMatrix(newSize, P1, P4, temp1);  
subtractMatrix(newSize, temp1, P5, temp2);  
addMatrix(newSize, temp2, P7, C11);
```

```
// C12 = P3 + P5  
addMatrix(newSize, P3, P5, C12);
```

```
// C21 = P2 + P4  
addMatrix(newSize, P2, P4, C21);
```

```
// C22 = P1 + P3 - P2 + P6  
addMatrix(newSize, P1, P3, temp1);  
subtractMatrix(newSize, temp1, P2, temp2);  
addMatrix(newSize, temp2, P6, C22);
```

```
// Combine C11, C12, C21, C22 into result matrix C
```

```

for (int i = 0; i < newSize; i++) {
    for (int j = 0; j < newSize; j++) {
        C[i][j] = C11[i][j];
        C[i][j + newSize] = C12[i][j];
        C[i + newSize][j] = C21[i][j];
        C[i + newSize][j + newSize] = C22[i][j];
    }
}

int main() {
    int n = 2; // Matrix size 2x2 (can be adjusted for larger sizes)

    int A[2][2] = { {1, 2}, {3, 4} };
    int B[2][2] = { {5, 6}, {7, 8} };
    int C[2][2]; // Result matrix

    strassenMultiply(n, A, B, C);

    // Print the result
    printf("Result of Strassen Matrix Multiplication:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", C[i][j]);
        }
    }
}

```

```
    }

    printf("\n");

}

return 0;
}
```

MERGE SORT

```
#include <stdio.h>

// Function to merge two halves of the array
void merge(int arr[], int left, int mid, int right) {

    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Temporary arrays to hold the left and right halves
    int leftArr[n1], rightArr[n2];

    // Copy data to temporary arrays
    for (i = 0; i < n1; i++)
        leftArr[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        rightArr[j] = arr[mid + 1 + j];
```

```
i = 0; // Initial index of the left array
j = 0; // Initial index of the right array
k = left; // Initial index of merged array

// Merge the two halves
while (i < n1 && j < n2) {
    if (leftArr[i] <= rightArr[j]) {
        arr[k] = leftArr[i];
        i++;
    } else {
        arr[k] = rightArr[j];
        j++;
    }
    k++;
}

// Copy the remaining elements of leftArr[], if any
while (i < n1) {
    arr[k] = leftArr[i];
    i++;
    k++;
}

// Copy the remaining elements of rightArr[], if any
```

```
while (j < n2) {
    arr[k] = rightArr[j];
    j++;
    k++;
}

// Function to implement Merge Sort
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Sort the first and second halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // Merge the sorted halves
        merge(arr, left, mid, right);
    }
}

// Function to print the array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
```

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

// Main function
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array: \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("Sorted array: \n");
    printArray(arr, arr_size);
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ code editor with the file 'MERGE SORT.cpp' open. The code implements a merge sort algorithm. A terminal window shows the execution of the program, printing the input array [12, 11, 13, 5, 6, 7] and the sorted array [5, 6, 7, 11, 12, 13]. Below the terminal is a 'Compiler' tab showing compilation results with 0 errors and 0 warnings, and a file size of 130.373046875 KiB.

```

57     // Merge the sorted halves
58     merge(arr, left, mid, right);
59 }
60 }
61 }
62 }
63 // Function to print the array
64 void printArray(int arr[], int size) {
65     for (int i = 0; i < size; i++) {
66         printf("%d ", arr[i]);
67         printf("\n");
68     }
69 }
70 // Main function
71 int main() {
72     int arr[] = {12, 11, 13, 5, 6, 7};
73     int arr_size = sizeof(arr) / sizeof(arr[0]);
74
75     printf("Given array: \n");
76     printArray(arr, arr_size);
77
78     mergeSort(arr, 0, arr_size - 1);
79
80     printf("Sorted array: \n");
81     printArray(arr, arr_size);
82     return 0;
83 }

```

MAX AND MIN

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

```
int main() {
```

```
    int n, i, max, min;
```

```
// Ask the user for the number of elements
```

```
printf("Enter the number of elements in the list: ");
```

```
scanf("%d", &n);
```

```
int arr[n]; // Declare an array of size n
```

```
// Input elements from the user
```

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

```
for (i = 0; i < n; i++) {  
    scanf("%d", &arr[i]);  
}  
  
// Initialize max and min to the first element of the array  
max = min = arr[0];  
  
// Loop through the array to find the max and min  
for (i = 1; i < n; i++) {  
    if (arr[i] > max) {  
        max = arr[i];  
    }  
    if (arr[i] < min) {  
        min = arr[i];  
    }  
}  
  
// Output the results  
printf("Maximum element: %d\n", max);  
printf("Minimum element: %d\n", min);  
  
return 0;  
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ code editor with the file 'MAX AND MIN.cpp' open. The code implements a program to find the maximum and minimum elements in an array of integers. A terminal window to the right shows the output of the program, which asks for the number of elements, lists them, and then outputs the maximum and minimum values. Below the editor is a 'Compiler' tab showing compilation results with 0 errors and 0 warnings.

```

10 int arr[n]; // Declare an array of size n
11
12 // Input elements from the user
13 printf("Enter the elements:\n");
14 for (i = 0; i < n; i++) {
15     scanf("%d", &arr[i]);
16 }
17
18 // Initialize max and min to the first element of the array
19 max = min = arr[0];
20
21 // Loop through the array to find the max and min
22 for (i = 1; i < n; i++) {
23     if (arr[i] > max) {
24         max = arr[i];
25     }
26     if (arr[i] < min) {
27         min = arr[i];
28     }
29 }
30
31 // Output the results
32 printf("Maximum element: %d\n", max);
33 printf("Minimum element: %d\n", min);
34
35 return 0;
36

```

PRIME NUMBERS IN LIMIT

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

```
#include <stdbool.h>
```

```
// Function to check if a number is prime
```

```
bool isPrime(int num) {
```

```
    if (num <= 1) return false;
```

```
    if (num == 2) return true; // 2 is the only even prime number
```

```
    if (num % 2 == 0) return false; // Eliminate even numbers
```

```
    for (int i = 3; i * i <= num; i += 2) {
```

```
        if (num % i == 0) return false;
```

```
}
```

```
return true;  
}  
  
int main() {  
    int numbers[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; // Example list of numbers  
    int size = sizeof(numbers) / sizeof(numbers[0]); // Calculate size of the array  
  
    printf("Prime numbers in the list:\n");  
    for (int i = 0; i < size; i++) {  
        if (isPrime(numbers[i])) {  
            printf("%d\n", numbers[i]);  
        }  
    }  
  
    return 0;  
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ source file named Untitled1 containing a function to check if a number is prime and another to print prime numbers from a list. The code uses standard input and output functions. A terminal window shows the execution of the program, which asks for a number, prints '4 is a non prime', and then exits. Below the terminal is a compilation results window showing no errors or warnings. The system tray at the bottom right indicates the date and time as 9/19/2024 at 2:19 PM.

```

C:\Users\khato\OneDrive\Documents\os\PRIME IN LIST.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug
Untitled1 perfect number using ifelse.cpp fibo using recursion.cpp armstrong.cpp
PALLINDROME.cpp COPY STRING.cpp BINARY SEARCH.cpp REVERSE STRING.cpp
1 #include <stdbool.h>
2
3
4 // Function to check if a number is prime
5 bool isPrime(int num) {
6     if (num <= 1) return false;
7     if (num == 2) return true; // 2 is the only even prime number
8     if (num % 2 == 0) return false; // Eliminate even numbers
9
10    for (int i = 3; i * i <= num; i += 2) {
11        if (num % i == 0) return false;
12    }
13
14    return true;
15
16 int main() {
17     int numbers[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; // Example
18     int size = sizeof(numbers) / sizeof(numbers[0]); // Calculate size
19
20     printf("Prime numbers in the list:\n");
21     for (int i = 0; i < size; i++) {
22         if (isPrime(numbers[i])) {
23             printf("%d\n", numbers[i]);
24         }
25     }
26
27     return 0;
28 }

```

PRIME NUMBERS UPTO

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

```
#include <stdbool.h>
```

```
// Function to check if a number is prime
```

```
bool isPrime(int num) {
```

```
    if (num <= 1) return false;
```

```
    if (num == 2) return true;
```

```
    if (num % 2 == 0) return false;
```

```
    for (int i = 3; i * i <= num; i += 2) {
```

```
        if (num % i == 0) return false;
```

```
}
```

```
        return true;  
    }  
  
int main() {  
    int limit;  
  
    // Get the limit from the user  
    printf("Enter the limit: ");  
    scanf("%d", &limit);  
  
    printf("Prime numbers up to %d are:\n", limit);  
  
    // Find and print all prime numbers up to the limit  
    for (int num = 2; num <= limit; num++) {  
        if (isPrime(num)) {  
            printf("%d ", num);  
        }  
    }  
  
    printf("\n");  
  
    return 0;  
}
```

```

C:\Users\khato\OneDrive\Documents\os\PRIME IN LIST.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug
Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp
PALLINDROME.cpp COPY STRING.cpp BINARY SEARCH.cpp REVERSE STRING.cpp
10 for (int i = 3; i * i <= num; i += 2) {
11     if (num % i == 0) return false;
12 }
13 return true;
14
15
16
17 int main() {
18     int limit;
19
20     // Get the limit from the user
21     printf("Enter the limit: ");
22     scanf("%d", &limit);
23
24     printf("Prime numbers up to %d are:\n", limit);
25
26     // Find and print all prime numbers up to the limit
27     for (int num = 2; num <= limit; num++) {
28         if (isPrime(num)) {
29             printf("%d ", num);
30         }
31     }
32
33     printf("\n");
34
35     return 0;
36 }

```

enter a number4
4 is a non prime
Process exited after 2.357 seconds with return value 0
Press any key to continue . . . |

return value 0

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\PRIME IN LIST.exe
- Output Size: 129.302734375 KiB
- Compilation Time: 0.25s

Line: 25 Col: 1 Sek 0 Lines: 37 Length: 745 Insert Done parsing in 0.031 seconds

KNAPSACK USING GREEDY TECHNIQUE

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

```
// Structure to represent an item
```

```
typedef struct {

    int weight;

    int value;

    double ratio;

} Item;
```

```
// Function to sort items based on their value-to-weight ratio in descending order
```

```
void sortItems(Item items[], int n) {
```

```
    Item temp;
```

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

```

for (int j = 0; j < n-i-1; j++) {
    if (items[j].ratio < items[j+1].ratio) {
        // Swap items[j] and items[j+1]
        temp = items[j];
        items[j] = items[j+1];
        items[j+1] = temp;
    }
}
}

// Function to solve the fractional knapsack problem
double fractionalKnapsack(Item items[], int n, int capacity) {
    sortItems(items, n);

    double totalValue = 0.0;
    int remainingCapacity = capacity;

    for (int i = 0; i < n; i++) {
        if (items[i].weight <= remainingCapacity) {
            // Take the whole item
            remainingCapacity -= items[i].weight;
            totalValue += items[i].value;
        } else {

```

```
// Take the fraction of the item  
totalValue += items[i].value * ((double)remainingCapacity /  
items[i].weight);  
  
break;  
}  
}  
  
return totalValue;  
}
```

```
int main() {  
    int n, capacity;  
  
    // Input number of items and knapsack capacity  
    printf("Enter the number of items: ");  
    scanf("%d", &n);  
  
    printf("Enter the capacity of the knapsack: ");  
    scanf("%d", &capacity);
```

```
Item items[n];
```

```
// Input weight and value of each item  
for (int i = 0; i < n; i++) {  
  
    printf("Enter weight and value for item %d: ", i + 1);  
    scanf("%d %d", &items[i].weight, &items[i].value);  
  
    items[i].ratio = (double)items[i].value / items[i].weight;
```

```
}
```

```
double maxValue = fractionalKnapsack(items, n, capacity);
```

```
printf("Maximum value in the knapsack = %.2f\n", maxValue);
```

```
return 0;
```

```
}
```

The screenshot shows the Dev-C++ IDE interface. The code editor window displays the C++ code for a fractional knapsack algorithm. The code includes input handling for the number of items and knapsack capacity, a loop to read item weights and values, and a call to the `fractionalKnapsack` function. The output window shows the execution of the program, where it prompts for input and then prints the maximum value found. The status bar at the bottom shows compilation results with 0 errors and 0 warnings, and the file path `C:\Users\khato\OneDrive\Documents\os\KNAPSACK GREEDY.exe`.

```
C:\Users\khato\OneDrive\Documents\os\KNAPSACK GREEDY.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
File Project Classes Debug
Untitled1 perfect number using else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp largest element.cpp factorial.cpp prime.cpp selection sort.cpp bubble sort.cpp MATRIX MUL.cpp PALLINDROME.cpp
COPY STRING.cpp BINARY SEARCH.cpp REVERSE STRING.cpp LENGTH OF STRING.cpp STRASSEN MATRIX.cpp MERGE SORT.cpp MAX AND MIN.cpp PRIME IN LIST.cpp KNAPSACK GREEDY.cpp
(globals)
43     return totalValue;
44 }
45
46 int main() {
47     int n, capacity;
48
49     // Input number of items and knapsack capacity
50     printf("Enter the number of items: ");
51     scanf("%d", &n);
52     printf("Enter the capacity of the knapsack: ");
53     scanf("%d", &capacity);
54
55     Item items[n];
56
57     // Input weight and value of each item
58     for (int i = 0; i < n; i++) {
59         printf("Enter weight and value for item %d: ", i + 1);
60         scanf("%d %d", &items[i].weight, &items[i].value);
61         items[i].ratio = (double)items[i].value / items[i].weight;
62     }
63
64     double maxValue = fractionalKnapsack(items, n, capacity);
65
66     printf("Maximum value in the knapsack = %.2f\n", maxValue);
67
68     return 0;
69 }

C:\Users\khato\OneDrive\Docx Enter the number of items: 4
Enter the capacity of the knapsack: 10
Enter weight and value for item 1: 2
3
Enter weight and value for item 2: 4
5
Enter weight and value for item 3: 6
7
Enter weight and value for item 4: 8
7
Maximum value in the knapsack = 12.67
-----
Process exited after 11.02 seconds with return value 0
Press any key to continue . . . |
```

MST USING GREEDY

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

```
#include <stdlib.h>
```

```
// Define the maximum number of vertices
```

```
#define MAX_VERTICES 100
```

```
#define INF 999999

// Structure to represent an edge
typedef struct {
    int src, dest, weight;
} Edge;

// Structure to represent a subset for Union-Find
typedef struct {
    int parent, rank;
} Subset;

// Function prototypes
int find(Subset subsets[], int i);
void unionSets(Subset subsets[], int x, int y);
int compare(const void *a, const void *b);

// Kruskal's algorithm to find MST
void kruskal(Edge edges[], int V, int E) {
    Edge result[V]; // To store the resulting MST
    Subset subsets[V]; // To represent the Union-Find structure
    int e = 0; // Count of edges in MST
    int i = 0; // Initial index of sorted edges
```

```

// Initialize subsets
for (int v = 0; v < V; ++v) {
    subsets[v].parent = v;
    subsets[v].rank = 0;
}

// Sort all edges by weight
qsort(edges, E, sizeof(edges[0]), compare);

// Process each edge in sorted order
while (e < V - 1 && i < E) {
    Edge next_edge = edges[i++];
    int x = find(subsets, next_edge.src);
    int y = find(subsets, next_edge.dest);

    // If including this edge does not cause a cycle
    if (x != y) {
        result[e++] = next_edge;
        unionSets(subsets, x, y);
    }
}

// Print the MST
printf("Edges in the MST:\n");

```

```
int mst_weight = 0;  
for (i = 0; i < e; ++i) {  
    printf("%d -- %d == %d\n", result[i].src, result[i].dest, result[i].weight);  
    mst_weight += result[i].weight;  
}  
printf("Weight of MST: %d\n", mst_weight);  
}
```

```
// Find function for Union-Find  
int find(Subset subsets[], int i) {  
    if (subsets[i].parent != i)  
        subsets[i].parent = find(subsets, subsets[i].parent);  
    return subsets[i].parent;  
}
```

```
// Union function for Union-Find  
void unionSets(Subset subsets[], int x, int y) {  
    int xroot = find(subsets, x);  
    int yroot = find(subsets, y);  
  
    if (subsets[xroot].rank < subsets[yroot].rank)  
        subsets[xroot].parent = yroot;  
    else if (subsets[xroot].rank > subsets[yroot].rank)  
        subsets[yroot].parent = xroot;
```

```

else {
    subsets[yroot].parent = xroot;
    subsets[xroot].rank++;
}

// Comparison function for sorting edges
int compare(const void *a, const void *b) {
    return ((Edge *)a)->weight - ((Edge *)b)->weight;
}

// Main function
int main() {
    int V, E;
    printf("Enter number of vertices and edges: ");
    scanf("%d %d", &V, &E);

    Edge edges[E];
    printf("Enter the edges in format src dest weight:\n");
    for (int i = 0; i < E; i++) {
        scanf("%d %d %d", &edges[i].src, &edges[i].dest, &edges[i].weight);
    }

    kruskal(edges, V, E);
}

```

```

    return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The main window displays the C++ code for 'MST GREEDY.cpp'. The code implements a greedy algorithm to find a Minimum Spanning Tree (MST) using Kruskal's algorithm. It includes functions for edge comparison and MST construction. The output window shows the program's execution, prompting for vertices and edges, listing the edges selected for the MST, and calculating the total weight.

```

C:\Users\khato\OneDrive\Documents\os\MST GREEDY.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp largest element.cpp factorial.cpp prime.cpp selection sort.cpp bubble sort.cpp MATRIX MUL.cpp PALLINDROME.cpp
COPY STRING.cpp BINARY SEARCH.cpp REVERSE STRING.cpp LENGTH OF STRING.cpp STRASSEN MATRIX.cpp MERGE SORT.cpp MAX AND MIN.cpp PRIME IN LIST.cpp KNAPSACK GREEDY.cpp MST GREEDY.cpp
(globals)
78     else {
79         subsets[yroot].parent = xroot;
80         subsets[xroot].rank++;
81     }
82 }
83
84 // Comparison function for sorting edges
85 int compare(const void *a, const void *b) {
86     return ((Edge *)a)->weight - ((Edge *)b)->weight;
87 }
88
89 // Main function
90 int main() {
91     int V, E;
92     printf("Enter number of vertices and edges: ");
93     scanf("%d %d", &V, &E);
94
95     Edge edges[E];
96     printf("Enter the edges in format src dest weight:\n");
97     for (int i = 0; i < E; i++) {
98         scanf("%d %d %d", &edges[i].src, &edges[i].dest, &edges[i].weight);
99     }
100
101     kruskal(edges, V, E);
102
103     return 0;
104 }
105

```

C:\Users\khato\OneDrive\Doc x + - Enter number of vertices and edges: 5
5
Enter the edges in format src dest weight:
5
6
7
8
1
2
3
4
5
6
7
6
6
54
3
2
Edges in the MST:
0 -- 1 == 2
54 -- 0 == 2
3 -- 4 == 5
Weight of MST: 9

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\MST GREEDY.exe
- Output Size: 131.5849609375 Kib
- Compilation Time: 0.23s

Line: 105 Col: 1 Sek: 0 Lines: 105 Length: 2852 Insert Done parsing in 0.016 seconds

OPTIMAL BINARY SEARCH TREE

```

#include <stdio.h>

#include <limits.h>

// Function prototypes

void printOptimalBST(int i, int j, int s[][j+1], int keys[], int n);

void optimalBST(int keys[], int freq[], int n);

int main() {

    int keys[] = {10, 12, 20}; // Keys in the BST

```

```

int freq[] = {34, 8, 50}; // Frequencies of the keys
int n = sizeof(keys)/sizeof(keys[0]);

optimalBST(keys, freq, n);

return 0;
}

// Function to print the optimal BST
void printOptimalBST(int i, int j, int s[][j+1], int keys[], int n) {
    if (i <= j) {
        printf("Key %d (root) with keys in range [%d-%d]\n", keys[s[i][j]], i, j);
        printOptimalBST(i, s[i][j] - 1, s, keys, n);
        printOptimalBST(s[i][j] + 1, j, s, keys, n);
    }
}

// Function to construct and print the optimal BST
void optimalBST(int keys[], int freq[], int n) {
    int i, j, len, r;
    int cost[n][n], s[n][n];

    // Initialize cost and s tables
    for (i = 0; i < n; i++) {

```

```

cost[i][i] = freq[i];
s[i][i] = i;
for (j = i + 1; j < n; j++) {
    cost[i][j] = INT_MAX;
}
}

// Build the cost and s tables
for (len = 2; len <= n; len++) {
    for (i = 0; i <= n - len; i++) {
        j = i + len - 1;
        int sum = 0;
        for (r = i; r <= j; r++) {
            sum += freq[r];
        }
        for (r = i; r <= j; r++) {
            int c = (r > i ? cost[i][r - 1] : 0) + (r < j ? cost[r + 1][j] : 0) + sum;
            if (c < cost[i][j]) {
                cost[i][j] = c;
                s[i][j] = r;
            }
        }
    }
}

```

```
}

// Print the cost of the optimal BST
printf("Cost of Optimal BST: %d\n", cost[0][n - 1]);

// Print the optimal BST structure
printf("Optimal BST structure:\n");
printOptimalBST(0, n - 1, s, keys, n);

}
```

BINOMIAL COEFFICIENT

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

```
// Function to calculate factorial of a number
unsigned long long factorial(int n) {
    if (n == 0 || n == 1) return 1;
    unsigned long long fact = 1;
    for (int i = 2; i <= n; i++) {
        fact *= i;
    }
    return fact;
}
```

```
// Function to calculate binomial coefficient
unsigned long long binomialCoefficient(int n, int k) {
```

```
if (k > n - k) k = n - k; // Take advantage of symmetry
unsigned long long numerator = 1;
unsigned long long denominator = 1;

for (int i = 0; i < k; i++) {
    numerator *= (n - i);
    denominator *= (i + 1);
}

return numerator / denominator;
}

int main() {
    int n, k;

    printf("Enter values for n and k: ");
    scanf("%d %d", &n, &k);

    if (k > n || n < 0 || k < 0) {
        printf("Invalid values. Ensure 0 <= k <= n and n >= 0.\n");
        return 1;
    }

    printf("Binomial Coefficient C(%d, %d) = %llu\n", n, k, binomialCoefficient(n, k));
}
```

```

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a file named `BINOMIAL COEFFICIENT.cpp` with the following content:

```

15     if (k > n - k) k = n - k; // Take advantage of symmetry
16     unsigned long long numerator = 1;
17     unsigned long long denominator = 1;
18
19     for (int i = 0; i < k; i++) {
20         numerator *= (n - i);
21         denominator *= (i + 1);
22     }
23
24     return numerator / denominator;
25 }
26
27 int main() {
28     int n, k;
29
30     printf("Enter values for n and k: ");
31     scanf("%d %d", &n, &k);
32
33     if (k > n || n < 0 || k < 0) {
34         printf("Invalid values. Ensure 0 <= k <= n and n >= 0");
35         return 1;
36     }
37
38     printf("Binomial Coefficient C(%d, %d) = %lu\n", n, k);
39
40     return 0;
41 }

```

The output window shows the program's execution:

```

Enter values for n and k: 5 3
Binomial Coefficient C(5, 3) = 10
Process exited after 1.837 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom indicates the current line (Line: 42), column (Col: 1), and other details.

REVERSE A NUMBER

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

```

int main() {

    int num, reversed = 0, remainder;

    // Input the number from the user
    printf("Enter an integer: ");
    scanf("%d", &num);

    // Reverse the number

```

```

while (num != 0) {

    remainder = num % 10;      // Get the last digit

    reversed = reversed * 10 + remainder; // Append the digit to reversed

    num /= 10;                // Remove the last digit

}

// Output the reversed number

printf("Reversed number: %d\n", reversed);

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays the C++ program for reversing a number. On the right, a terminal window shows the execution of the program. The terminal output includes the input 'Enter an integer: 432', the reversed number 'Reversed number: 234', and the message 'Process exited after 2.351 seconds with return value 0'. Below the terminal is a 'Compiler' window showing compilation results with 0 errors and 0 warnings, and an output size of 128.1015625 Kib.

```

C:\Users\khato\OneDrive\Documents\os\REVERSE NUMBER.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
[Icons] (globals)
Project Classes Debug Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp gcd.cpp largest element.cpp factorial.cpp prime.cpp selection sort.cpp bubble sort.cpp MATRIX MUL.cpp PALLINDROME.cpp COPY STRING.cpp BINARY SEARCH.cpp REVERSE STRING.cpp LENGTH OF STRING.cpp STRASSEN MATRIX.cpp MERGE SORT.cpp MAX AND MIN.cpp PRIME IN LIST.cpp KNAKPSACK GREEDY.cpp MST GREEDY.cpp OBST.cpp BINOMIAL COEFFICIENT.cpp REVERSE NUMBER.cpp
1 #include <stdio.h>
2
3 int main() {
4     int num, reversed = 0, remainder;
5
6     // Input the number from the user
7     printf("Enter an integer: ");
8     scanf("%d", &num);
9
10    // Reverse the number
11    while (num != 0) {
12        remainder = num % 10;          // Get the last digit
13        reversed = reversed * 10 + remainder; // Append the digit to reversed
14        num /= 10;                  // Remove the last digit
15    }
16
17    // Output the reversed number
18    printf("Reversed number: %d\n", reversed);
19
20    return 0;
21
22 }

C:\Users\khato\OneDrive\Docx Enter an integer: 432
Reversed number: 234
-----
Process exited after 2.351 seconds with return value
e 0
Press any key to continue . . .

```

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\REVERSE NUMBER.exe
- Output Size: 128.1015625 Kib
- Compilation Time: 0.22s

Line: 22 Col: 1 Sek: 0 Lines: 22 Length: 544 Insert Done parsing in 0.015 seconds

PERFECT NUMBER

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

```
int main() {
    int num, i, sum = 0;

    // Input number from user
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    // Check if the number is positive
    if (num <= 0) {
        printf("The number must be a positive integer.\n");
        return 1; // Exit the program with error code
    }

    // Find sum of proper divisors
    for (i = 1; i <= num / 2; ++i) {
        if (num % i == 0) {
            sum += i;
        }
    }

    // Check if the number is a perfect number
    if (sum == num) {
        printf("%d is a perfect number.\n", num);
    } else {
```

```

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

    }

return 0; // Exit the program successfully
}

```

```

C:\Users\khato\OneDrive\Documents\os\PERFECT NUMBER.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASTyle Window Help
Project Classes Debug Untitled1 perfect number using if else.cpp fibo using recursion.cpp armstrong.cpp
PALLINDROME.cpp COPY STRING.cpp BINARY SEARCH.cpp REVERSE STRING.cpp
KNAPSACK GREEDY.cpp MST GREEDY.cpp OBST.cpp
6 // Input number from user
7 printf("Enter a positive integer: ");
8 scanf("%d", &num);
9
10 // Check if the number is positive
11 if (num <= 0) {
12     printf("The number must be a positive integer.\n");
13     return 1; // Exit the program with error code
14 }
15
16 // Find sum of proper divisors
17 for (i = 1; i <= num / 2; ++i) {
18     if (num % i == 0) {
19         sum += i;
20     }
21 }
22
23 // Check if the number is a perfect number
24 if (sum == num) {
25     printf("%d is a perfect number.\n", num);
26 } else {
27     printf("%d is not a perfect number.\n", num);
28 }
29
30 return 0; // Exit the program successfully
31
32

```

Enter a positive integer: 6
6 is a perfect number.
Process exited after 2.367 seconds with return value 0
Press any key to continue . . .

TSP USING DYNAMIC PROGRAMMING

```

#include <stdio.h>

#include <limits.h>

#define MAX 16

#define INF 1000000

int dist[MAX][MAX];

int dp[1 << MAX][MAX];

```

```
// Function to find the minimum of two numbers
int min(int a, int b) {
    return (a < b) ? a : b;
}

// Recursive function to solve TSP
int tsp(int mask, int pos, int n) {
    // If all cities have been visited, return to the starting point
    if (mask == (1 << n) - 1) {
        return dist[pos][0];
    }

    // If this state has already been computed
    if (dp[mask][pos] != -1) {
        return dp[mask][pos];
    }

    int answer = INF;

    // Try to visit all unvisited cities
    for (int city = 0; city < n; city++) {
        // If city is not visited
        if ((mask & (1 << city)) == 0) {
```

```
int newAnswer = dist[pos][city] + tsp(mask | (1 << city), city, n);
answer = min(answer, newAnswer);

}

}

// Store and return the result
dp[mask][pos] = answer;
return answer;

}

int main() {
    int n;

    printf("Enter the number of cities: ");
    scanf("%d", &n);

    printf("Enter the distance matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &dist[i][j]);
        }
    }

    // Initialize the dp table
```

```

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

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

        dp[i][j] = -1;

    }

}

// Start from city 0, with only city 0 visited

int result = tsp(1, 0, n);

printf("The minimum cost is: %d\n", result);

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The project navigation bar at the top lists several files: perfect number using if else.cpp, fibo using recursion.cpp, armstrong.cpp, gcd.cpp, PALLINDROME.cpp, COPY STRING.cpp, BINARY SEARCH.cpp, REVERSE STRING.cpp, LEAP YEAR.cpp, KNAPSACK GREEDY.cpp, MST GREEDY.cpp, OBST.cpp, BINOMIAL COEFFICIENTS.cpp, bubble sort.cpp, MATRIX MUL.cpp, AX AND MIN.cpp, PRIME IN LIST.cpp, R.cpp, and TSP USING DP.cpp.

The main code editor window displays the C++ source code for "TSP USING DP.cpp". The code includes functions for reading input, initializing a distance matrix, and calculating the minimum cost using dynamic programming. It also includes comments explaining the steps: starting from city 0 with only city 0 visited, and printing the result.

The terminal window shows the execution of the program. It prompts the user for the number of cities (4) and the distance matrix (4x4 matrix). The output shows the minimum cost is 12.

The status bar at the bottom provides compilation information: Compilation results..., Errors: 0, Warnings: 0, Output Filename: C:\Users\khato\OneDrive\Documents\os\TSP USING DP.exe, Output Size: 129.8505859375 Kib, and Compilation Time: 0.20s.

PATTERN

```
#include <stdio.h>

int main() {
    int height, i, j;

    // Ask the user for the height of the pyramid
    printf("Enter the height of the pyramid: ");
    scanf("%d", &height);

    // Loop to print each level of the pyramid
    for (i = 1; i <= height; i++) {
        // Print spaces for the left side of the pyramid
        for (j = i; j < height; j++) {
            printf(" ");
        }

        // Print stars for the pyramid body
        for (j = 1; j <= (2 * i - 1); j++) {
            printf("*");
        }

        // Move to the next line after each level
        printf("\n");
    }

    return 0;
}
```

}

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C++ program named Untitled1 that prints a pyramid pattern based on user input. The terminal window shows the program's output for an input of 4, displaying a pyramid of stars. The compiler log shows a successful compilation with no errors or warnings.

```

1 #include <stdio.h>
2
3 int main() {
4     int height, i, j;
5
6     // Ask the user for the height of the pyramid
7     printf("Enter the height of the pyramid: ");
8     scanf("%d", &height);
9
10    // Loop to print each Level of the pyramid
11    for (i = 1; i <= height; i++) {
12        // Print spaces for the left side of the pyramid
13        for (j = 1; j < i; j++) {
14            printf(" ");
15        }
16        // Print stars for the pyramid body
17        for (j = 1; j <= (2 * i - 1); j++) {
18            printf("*");
19        }
20        // Move to the next line after each level
21        printf("\n");
22    }
23
24    return 0;
25
26

```

Compiler Results:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\os\PATTERN.exe
- Output Size: 128.7734375 Kib
- Compilation Time: 0.27s

FLOYD ALGORITHM

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

```
#define INF 99999 // A large value to represent infinity
```

```
#define V 4      // Number of vertices in the graph
```

```
// Function to print the solution matrix
```

```
void printSolution(int dist[][V]) {
```

```
    printf("The following matrix shows the shortest distances between every pair of
vertices:\n");
```

```
    for (int i = 0; i < V; i++) {
```

```
        for (int j = 0; j < V; j++) {
```

```
            if (dist[i][j] == INF)
```

```
                printf("%7s", "INF");
```

```

        else
            printf("%7d", dist[i][j]);
    }
    printf("\n");
}
}

// Floyd-Warshall algorithm to find the shortest path between all pairs of vertices
void floydWarshall(int graph[][V]) {
    int dist[V][V], i, j, k;

    // Initialize the solution matrix the same as the input graph matrix
    for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
            dist[i][j] = graph[i][j];

    // Adding vertices individually to the set of intermediate vertices
    for (k = 0; k < V; k++) {
        for (i = 0; i < V; i++) {
            for (j = 0; j < V; j++) {
                // Update dist[i][j] if a shorter path exists through vertex k
                if (dist[i][k] + dist[k][j] < dist[i][j])
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }
}

```

```
}

// Print the final shortest distance matrix
printSolution(dist);

}

int main() {
    // Input graph represented by an adjacency matrix
    int graph[V][V] = { {0, 3, INF, 7},
                        {8, 0, 2, INF},
                        {5, INF, 0, 1},
                        {2, INF, INF, 0} };

    // Execute the Floyd-Warshall algorithm
    floydWarshall(graph);

    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays the C++ code for the Floyd-Warshall algorithm. The code initializes a distance matrix, iterates through vertices to add them to the set of intermediate vertices, and prints the final shortest distance matrix. A terminal window shows the execution output, which includes a 4x4 distance matrix and a message indicating the process exited successfully.

```

19
20 // Floyd-Warshall algorithm to find the shortest path between all pairs of
21 void floyd_marshall(int graph[V][V]) {
22     int dist[V][V], i, j, k;
23
24     // Initialize the solution matrix the same as the input graph matrix
25     for (i = 0; i < V; i++)
26         for (j = 0; j < V; j++)
27             dist[i][j] = graph[i][j];
28
29     // Adding vertices individually to the set of intermediate vertices
30     for (k = 0; k < V; k++) {
31         for (i = 0; i < V; i++) {
32             for (j = 0; j < V; j++) {
33                 // Update dist[i][j] if a shorter path exists through vert
34                 if (dist[i][k] + dist[k][j] < dist[i][j])
35                     dist[i][j] = dist[i][k] + dist[k][j];
36             }
37         }
38     }
39
40     // Print the final shortest distance matrix
41     printSolution(dist);
42 }
43
44 int main() {
45     // Input graph represented by an adjacency matrix
46     int graph[V][V] = { { 0, 3, INF, 6 },
47                         { 3, 0, 2, 1 },
48                         { 2, 5, 0, 0 },
49                         { 0, 0, 0, 0 } };

```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\DAAs\FLOYD.exe
- Output Size: 130.3523390625 Kib
- Compilation Time: 0.30s

PASCAL TRIANGLE

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

```
// Function to calculate factorial of a number
```

```
long long factorial(int n) {

    long long fact = 1;

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

        fact *= i;

    }

    return fact;
}
```

```
// Function to print Pascal's triangle
```

```
void printPascalsTriangle(int rows) {

    for (int i = 0; i < rows; i++) {
```

```
// Print spaces for formatting
for (int j = 0; j < rows - i - 1; j++) {
    printf(" ");
}

// Calculate and print values of each row
for (int j = 0; j <= i; j++) {
    long long value = factorial(i) / (factorial(j) * factorial(i - j));
    printf("%4lld ", value);
}
printf("\n");

}

int main() {
    int rows;
    printf("Enter the number of rows for Pascal's triangle: ");
    scanf("%d", &rows);
    printPascalsTriangle(rows);
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays the source code for a program named PASCAL TRIANGLE.cpp. The code includes a factorial function, a printPascalsTriangle function that generates a Pascal triangle for a given number of rows, and a main function that prompts the user for input and calls the triangle printing function. A terminal window shows the execution of the program, where it asks for the number of rows (4) and then prints the first four rows of the Pascal triangle. The bottom panel shows the compiler log with no errors or warnings.

```

C:\Users\khato\OneDrive\Documents\DAAPASCAL TRIANGLE.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug FLOYD.cpp PASCAL TRIANGLE.cpp
8 }
9 return fact;
10 }
11
12 // Function to print Pascal's triangle
13 void printPascalsTriangle(int rows) {
14     for (int i = 0; i < rows; i++) {
15         // Print spaces for formatting
16         for (int j = 0; j < rows - i - 1; j++) {
17             printf(" ");
18         }
19
20         // Calculate and print values of each row
21         for (int j = 0; j <= i; j++) {
22             long long value = factorial(i) / (factorial(j) * factorial(i - j));
23             printf("%4lld ", value);
24         }
25         printf("\n");
26     }
27 }
28
29 int main() {
30     int rows;
31     printf("Enter the number of rows for Pascal's triangle: ");
32     scanf("%d", &rows);
33     printPascalsTriangle(rows);
34     return 0;
35 }

C:\Users\khato\OneDrive\Doc X + - ×
Enter the number of rows for Pascal's triangle: 4
      1
     1 1
    1 2 1
   1 3 3 1
Process exited after 2.136 seconds with return value 0
Press any key to continue . . .

```

OPTIMAL COST USING KNAPSACK

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

```
// Function to find the maximum of two numbers
```

```
int max(int a, int b) {
    return (a > b) ? a : b;
}
```

```
// Function to solve the 0/1 Knapsack problem
```

```
int knapsack(int capacity, int weights[], int values[], int n) {
    int i, w;
    int dp[n + 1][capacity + 1]; // Create a 2D DP array
```

```
// Build the DP table in a bottom-up manner
```

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

```

for (w = 0; w <= capacity; w++) {
    if (i == 0 || w == 0) {
        dp[i][w] = 0; // Base case: no items or zero capacity
    } else if (weights[i - 1] <= w) {
        dp[i][w] = max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);
    } else {
        dp[i][w] = dp[i - 1][w];
    }
}

// Return the maximum value that can be put in the knapsack
return dp[n][capacity];
}

int main() {
    int n, capacity;

    // Input number of items and capacity of knapsack
    printf("Enter the number of items: ");
    scanf("%d", &n);
    printf("Enter the capacity of the knapsack: ");
    scanf("%d", &capacity);

    int values[n], weights[n];
}

```

```
// Input the values and weights of the items
printf("Enter the values of the items: \n");
for (int i = 0; i < n; i++) {
    scanf("%d", &values[i]);
}

printf("Enter the weights of the items: \n");
for (int i = 0; i < n; i++) {
    scanf("%d", &weights[i]);
}

// Call the knapsack function and print the optimal cost
int optimal_cost = knapsack(capacity, weights, values, n);
printf("The optimal cost is: %d\n", optimal_cost);

return 0;
}
```

```

C:\Users\khato\OneDrive\Documents\DA\OBST USING KNAKPSACK.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug
FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNAKPSACK.cpp
30 int main() {
31     int n, capacity;
32
33     // Input number of items and capacity of knapsack
34     printf("Enter the number of items: ");
35     scanf("%d", &n);
36     printf("Enter the capacity of the knapsack: ");
37     scanf("%d", &capacity);
38
39     int values[n], weights[n];
40
41     // Input the values and weights of the items
42     printf("Enter the values of the items: \n");
43     for (int i = 0; i < n; i++) {
44         scanf("%d", &values[i]);
45     }
46
47     printf("Enter the weights of the items: \n");
48     for (int i = 0; i < n; i++) {
49         scanf("%d", &weights[i]);
50     }
51
52     // Call the knapsack function and print the optimal cost
53     int optimal_cost = knapsack(capacity, weights, values, n);
54     printf("The optimal cost is: %d\n", optimal_cost);
55
56     return 0;
57 }

-----  

Process exited after 7.862 seconds with retu  

rn value 0  

Press any key to continue . . .

```

Compiler Resources Compile Log Debug Find Results Close

Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\DA\OBST USING KNAKPSACK.exe
- Output Size: 130.3466796875 Kib
- Compilation Time: 0.22s

Line: 58 Cob: 1 Sek: 0 Lines: 58 Length: 1670 Insert Done parsing in 0.016 seconds

SUM OF DIGITS

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

```

int main() {

    int num, sum = 0, remainder;

    // Input a number from user
    printf("Enter an integer: ");

    scanf("%d", &num);

    // Loop to find the sum of digits
    while (num != 0) {

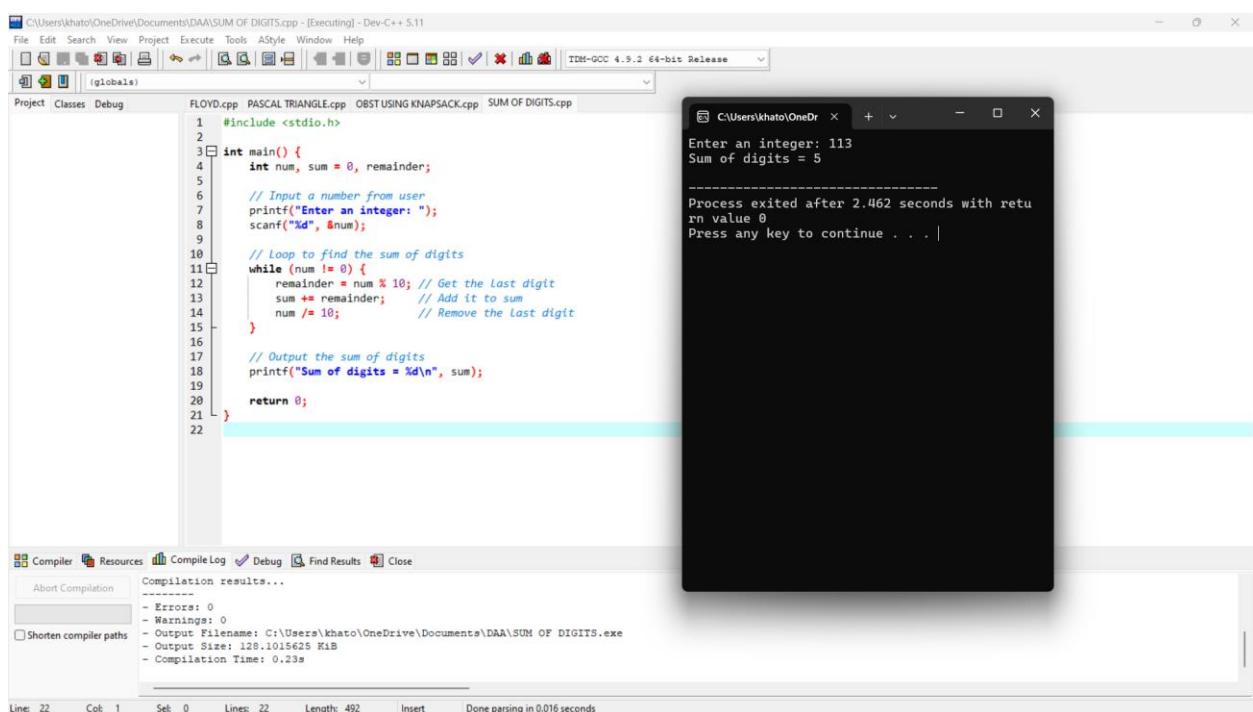
        remainder = num % 10; // Get the last digit

        sum += remainder; // Add it to sum

        num /= 10; // Remove the last digit
    }
}
```

```
}
```

```
// Output the sum of digits  
printf("Sum of digits = %d\n", sum);  
  
return 0;  
}
```



The screenshot shows the Dev-C++ IDE interface. The main window displays a C program to calculate the sum of digits of an integer. The code uses a while loop to repeatedly extract the last digit of the input number and add it to a sum until the number becomes zero. The output window shows the execution of the program, where it prompts the user for an integer (113) and then prints the sum of digits (5). The status bar at the bottom indicates the compilation results, showing 0 errors and 0 warnings, and provides file statistics like the output filename (C:\Users\khato\OneDrive\Documents\DAAA\SUM OF DIGITS.exe), size (128.1015625 Kib), and compilation time (0.23s).

```
#include <stdio.h>  
  
int main() {  
    int num, sum = 0, remainder;  
  
    // Input a number from user  
    printf("Enter an integer: ");  
    scanf("%d", &num);  
  
    // Loop to find the sum of digits  
    while (num != 0) {  
        remainder = num % 10; // Get the last digit  
        sum += remainder; // Add it to sum  
        num /= 10; // Remove the last digit  
    }  
  
    // Output the sum of digits  
    printf("Sum of digits = %d\n", sum);  
}  
  
return 0;
```

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

Project Classes Debug FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNAPSACK.cpp SUM OF DIGITS.cpp

Enter an integer: 113
Sum of digits = 5

Process exited after 2.462 seconds with return value 0
Press any key to continue . . . |

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\DAAA\SUM OF DIGITS.exe
- Output Size: 128.1015625 Kib
- Compilation Time: 0.23s

Line: 22 Col: 1 Sel: 0 Lines: 22 Length: 492 Insert Done parsing in 0.016 seconds

MAX ND MIN IN LIST

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

```
int main() {  
    int n, i;  
  
    int max, min;
```

```
// Ask the user for the number of elements
```

```
printf("Enter the number of elements in the array: ");
scanf("%d", &n);

int arr[n];

// Ask the user to enter the elements of the array
printf("Enter %d elements: \n", n);
for (i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
}

// Initialize max and min with the first element
max = min = arr[0];

// Loop through the array to find max and min
for (i = 1; i < n; i++) {
    if (arr[i] > max) {
        max = arr[i];
    }
    if (arr[i] < min) {
        min = arr[i];
    }
}

// Output the max and min
```

```

printf("Maximum element: %d\n", max);
printf("Minimum element: %d\n", min);

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The main window displays the C++ code for finding the maximum and minimum elements in an array. The terminal window shows the program's output: it asks for the number of elements, then lists them, and finally prints the maximum and minimum values. The compiler log window shows the compilation results, indicating 0 errors and 0 warnings.

```

C:\Users\khato\OneDrive\Documents\DAAlMAX ND MIN LIST.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNPASACK.cpp SUM OF DIGITS.cpp MAX ND MIN LIST.cpp
10
11     int arr[n];
12
13     // Ask the user to enter the elements of the array
14     //printf("Enter %d elements: \n", n);
15     for (i = 0; i < n; i++) {
16         scanf("%d", &arr[i]);
17     }
18
19     // Initialize max and min with the first element
20     max = min = arr[0];
21
22     // Loop through the array to find max and min
23     for (i = 1; i < n; i++) {
24         if (arr[i] > max) {
25             max = arr[i];
26         }
27         if (arr[i] < min) {
28             min = arr[i];
29         }
30     }
31
32     // Output the max and min
33     printf("Maximum element: %d\n", max);
34     printf("Minimum element: %d\n", min);
35
36     return 0;
37 }

// C:\Users\khato\OneDrive\Documents\DAAlMAX ND MIN LIST.exe
Enter the number of elements in the array: 3
Enter 3 elements:
4
5
18
Maximum element: 18
Minimum element: 4

Process exited after 4.394 seconds with return value 0
Press any key to continue . . .

```

N QUEENS USING BACKTRACKING

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

```
#include <stdbool.h>
```

```
#define N 8 // You can change this value for a different size board
```

```

// Function to print the solution
void printSolution(int board[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {

```

```

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

printf("\n");
}

}

// Function to check if a queen can be placed on board[row][col]
bool isSafe(int board[N][N], int row, int col) {
    int i, j;

    // Check this row on left side
    for (i = 0; i < col; i++)
        if (board[row][i])
            return false;

    // Check upper diagonal on left side
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;

    // Check lower diagonal on left side
    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;
}

```

```

    return true;
}

// Function to solve the N-Queens problem using backtracking
bool solveNQueensUtil(int board[N][N], int col) {
    // Base case: If all queens are placed, return true
    if (col >= N)
        return true;

    // Consider this column and try placing this queen in all rows one by one
    for (int i = 0; i < N; i++) {
        // Check if the queen can be placed on board[i][col]
        if (isSafe(board, i, col)) {
            // Place the queen
            board[i][col] = 1;

            // Recur to place the rest of the queens
            if (solveNQueensUtil(board, col + 1))
                return true;

            // If placing queen in board[i][col] doesn't lead to a solution, remove the
            // queen
            board[i][col] = 0; // BACKTRACK
        }
    }
}

```

```
// If the queen cannot be placed in any row in this column, return false
return false;

}

// Function to solve the N-Queens problem
bool solveNQueens() {
    int board[N][N] = {0}; // Initialize the board with 0s

    if (solveNQueensUtil(board, 0) == false) {
        printf("Solution does not exist\n");
        return false;
    }

    printSolution(board);
    return true;
}

// Driver code
int main() {
    solveNQueens();
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays the C++ code for solving the N-Queens problem. The code includes utility functions for backtracking and printing solutions, and a driver main function that calls the solver. The output window shows the 8x8 board configuration for an 8-queens solution, followed by a message indicating the process exited after 0.05344 seconds. The compiler log window shows no errors or warnings during compilation.

```

54     // If placing queen in board[i][col] doesn't lead to a solution, remove the queen
55     board[i][col] = 0; // BACKTRACK
56   }
57 }
58 // If the queen cannot be placed in any row in this column, return false;
59 }
60 // Function to solve the N-Queens problem
61 bool solveNQueens() {
62   int board[N][N] = {0}; // Initialize the board with 0s
63   if (solveNQueensUtil(board, 0) == false) {
64     printf("Solution does not exist\n");
65     return false;
66   }
67   printSolution(board);
68   return true;
69 }
70 // Driver code
71 int main() {
72   solveNQueens();
73   return 0;
74 }
75
76
77
78
79
80
81
82

```

33. INSERT A NUMBER IN LIST

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

```
void insertNumber(int arr[], int *size, int number) {
```

```
    int i, j;
```

```
// Find the correct position for the number
```

```
for (i = 0; i < *size; i++) {
```

```
    if (arr[i] > number) {
```

```
        break;
```

```
}
```

```
}
```

```
// Shift elements to the right to make space for the new number
```

```
for (j = *size; j > i; j--) {
```

```
    arr[j] = arr[j - 1];  
}  
  
// Insert the number  
arr[i] = number;  
(*size)++; // Increment the size of the array  
}  
  
int main() {  
    int arr[100], size, i, number;  
  
    // Get the size of the array  
    printf("Enter the number of elements in the array: ");  
    scanf("%d", &size);  
  
    // Get the elements of the array  
    printf("Enter %d elements in sorted order: \n", size);  
    for (i = 0; i < size; i++) {  
        scanf("%d", &arr[i]);  
    }  
  
    // Get the number to insert  
    printf("Enter the number to insert: ");  
    scanf("%d", &number);
```

```

// Insert the number in the correct position
insertNumber(arr, &size, number);

// Print the updated array
printf("Array after insertion: \n");
for (i = 0; i < size; i++) {
    printf("%d ", arr[i]);
}
printf("\n");

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The code editor window displays the C++ source code for inserting a number into an array. The code includes comments explaining the steps: getting the array size, inputting elements, getting the number to insert, inserting it, and printing the updated array. The terminal window shows the execution of the program, where the user enters 4 as the array size, followed by four sorted integers (1, 3, 5, 6). Then, the user enters 8 as the number to insert. The program outputs the original array followed by the inserted value. The status bar at the bottom provides compilation results, including the output file name (C:\Users\khato\OneDrive\Documents\DAA\INSERT NUMBER.exe), output size (130.4814453125 Kib), and compilation time (0.23s).

```

C:\Users\khato\OneDrive\Documents\DAAl\INSERT NUMBER.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASyntax Window Help
Project Classes Debug FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNAPSACK.cpp SUM OF DIGITS.cpp MAX ND MIN L...
(globals) 
24 int arr[100], size, i, number;
25
26 // Get the size of the array
27 printf("Enter the number of elements in the array: ");
28 scanf("%d", &size);
29
30 // Get the elements of the array
31 printf("Enter %d elements in sorted order: \n", size);
32 for (i = 0; i < size; i++) {
33     scanf("%d", &arr[i]);
34 }
35
36 // Get the number to insert
37 printf("Enter the number to insert: ");
38 scanf("%d", &number);
39
40 // Insert the number in the correct position
41 insertNumber(arr, &size, number);
42
43 // Print the updated array
44 printf("Array after insertion: \n");
45 for (i = 0; i < size; i++) {
46     printf("%d ", arr[i]);
47 }
48 printf("\n");
49
50 return 0;
51 }

Enter the number of elements in the array: 4
Enter 4 elements in sorted order:
1
3
5
6
Enter the number to insert: 8
Array after insertion:
1 3 5 6 8
-----
Process exited after 10.52 seconds with return value 0
Press any key to continue . . .

```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\DAAl\INSERT NUMBER.exe
- Output Size: 130.4814453125 Kib
- Compilation Time: 0.23s

Line: 52 Col: 1 Sel: 0 Lines: 52 Length: 1228 Insert Done parsing in 0 seconds

34. SUM OF SUBSETS

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

```

// Function to print a subset
void printSubset(int subset[], int size) {
    int sum = 0;
    printf("{ ");
    for (int i = 0; i < size; i++) {
        printf("%d ", subset[i]);
        sum += subset[i];
    }
    printf("} -> Sum = %d\n", sum);
}

// Recursive function to find the subsets
void findSubsets(int set[], int subset[], int n, int subsetSize, int index) {
    if (index == n) {
        printSubset(subset, subsetSize);
        return;
    }

    // Including set[index] in the current subset
    subset[subsetSize] = set[index];
    findSubsets(set, subset, n, subsetSize + 1, index + 1);

    // Not including set[index] in the current subset
    findSubsets(set, subset, n, subsetSize, index + 1);
}

```

```
int main() {
    int n;

    printf("Enter the number of elements in the set: ");
    scanf("%d", &n);

    int set[n], subset[n];

    printf("Enter the elements of the set:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &set[i]);
    }

    printf("The subsets and their sums are:\n");
    findSubsets(set, subset, n, 0, 0);

    return 0;
}
```

```

C:\Users\khato\OneDrive\Documents\DAASUM OF SUBSETS.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNAPSACK.cpp SUM OF DIGITS.cpp MAX ND MIN LIST.cpp N QUEENS.cpp INSERT NUMBER.cpp SUM OF SUBSETS.cpp
Project Classes Debug
19 }
20 // Including set[index] in the current subset
21 subset[subsetSize] = set[index];
22 findSubsets(set, subset, n, subsetSize + 1, index + 1);
23
24 // Not including set[index] in the current subset
25 findSubsets(set, subset, n, subsetSize, index + 1);
26
27 }
28
29 int main() {
30     int n;
31
32     printf("Enter the number of elements in the set: ");
33     scanf("%d", &n);
34
35     int set[n], subset[n];
36
37     printf("Enter the elements of the set:\n");
38     for (int i = 0; i < n; i++) {
39         scanf("%d", &set[i]);
40     }
41
42     printf("The subsets and their sums are:\n");
43     findSubsets(set, subset, n, 0, 0);
44
45     return 0;
46 }

```

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\DAASUM OF SUBSETS.exe
- Output Size: 129.8466796875 KiB
- Compilation Time: 0.24s

Line: 47 Col: 1 Sek 0 Lines: 47 Length: 1149 Insert Done parsing in 0.015 seconds

35. GRAPH COLORING

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

```
#include <stdbool.h>
```

```
#define V 4 // Number of vertices
```

```
// Function to print the solution
```

```
void printSolution(int color[]) {
```

```
    printf("Solution exists: Following are the assigned colors:\n");
```

```
    for (int i = 0; i < V; i++)
```

```
        printf("Vertex %d ---> Color %d\n", i, color[i]);
```

```
}
```

```
// Function to check if the current color assignment is safe
```

```
bool isSafe(int v, bool graph[V][V], int color[], int c) {
```

```

for (int i = 0; i < V; i++) {
    if (graph[v][i] && c == color[i]) // If adjacent and color is the same
        return false;
}
return true;
}

```

```

// A recursive utility function to solve the graph coloring problem
bool graphColoringUtil(bool graph[V][V], int m, int color[], int v) {
    // Base case: If all vertices are assigned a color then return true
    if (v == V)
        return true;

```

// Consider this vertex v and try different colors

```

for (int c = 1; c <= m; c++) {
    // Check if assignment of color c to v is valid
    if (isSafe(v, graph, color, c)) {
        color[v] = c;

```

// Recursively assign colors to the rest of the vertices

```

        if (graphColoringUtil(graph, m, color, v + 1))
            return true;

```

// If assigning color c doesn't lead to a solution, remove it

```

        color[v] = 0;
    }
}
```

```
    }

}

// If no color can be assigned, return false
return false;

}

// Function to solve the m Coloring problem
bool graphColoring(bool graph[V][V], int m) {
    // Initialize all color values as 0 (unassigned)
    int color[V] = {0};

    // Call graphColoringUtil() for vertex 0
    if (graphColoringUtil(graph, m, color, 0) == false) {
        printf("Solution does not exist\n");
        return false;
    }

    // Print the solution
    printSolution(color);
    return true;
}

int main() {
    // Example graph (4 vertices)
```

```

bool graph[V][V] = {
    {0, 1, 1, 1},
    {1, 0, 1, 0},
    {1, 1, 0, 1},
    {1, 0, 1, 0},
};

int m = 3; // Number of colors

// Check if the solution exists
graphColoring(graph, m);

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface with the following details:

- Editor Area:** Displays the C++ source code for `GRAPH COLORING.cpp`.
- Terminal Window:** Shows the program's output:


```

Solution exists: Following are the assigned colors:
Vertex 0 ---> Color 1
Vertex 1 ---> Color 2
Vertex 2 ---> Color 3
Vertex 3 ---> Color 2

Process exited after 0.04971 seconds with return value 0
Press any key to continue . . .
      
```
- Compiler Log:** Shows compilation results with 0 errors and 0 warnings.
- Status Bar:** Shows the current line (Line: 79), column (Col: 1), selection (Sel: 0), and other status information.

36. CONTAINER LOADER

```
#include <stdio.h>

#define MAX_ITEMS 100
#define MAX_CONTAINERS 100

void firstFit(int items[], int n, int containerCapacity) {
    int containers[MAX_CONTAINERS] = {0}; // Array to store remaining
    capacity of containers
    int containerCount = 0;

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

        // Try to fit the item in an existing container
        for (j = 0; j < containerCount; j++) {
            if (containers[j] >= items[i]) {
                containers[j] -= items[i];
                printf("Item %d of weight %d placed in container %d\n", i+1, items[i],
j+1);
                break;
            }
        }

        // If it doesn't fit in any existing container, use a new one
        if (j == containerCount) {
            containers[containerCount] = containerCapacity - items[i];
```

```

        printf("Item %d of weight %d placed in new container %d\n", i+1, items[i],
containerCount+1);

        containerCount++;

    }

}

printf("\nTotal containers used: %d\n", containerCount);

}

int main() {

    int items[MAX_ITEMS];
    int n, containerCapacity;

    // Input: number of items and container capacity
    printf("Enter the number of items: ");
    scanf("%d", &n);

    printf("Enter the container capacity: ");
    scanf("%d", &containerCapacity);

    // Input: weight of each item
    printf("Enter the weights of the items:\n");
    for (int i = 0; i < n; i++) {

        printf("Item %d: ", i+1);
        scanf("%d", &items[i]);

    }
}
```

```

// Call the function to find the container loading solution
firstFit(items, n, containerCapacity);

return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The left pane displays the code for `CONTAINER LOADER.cpp`. The right pane shows the terminal window where the program is running. The terminal output is as follows:

```

C:\Users\khato\OneDrive\Documents\DAACONTAINER LOADER.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools Style Window Help
Project Classes Debug FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNAKSPACK.cpp SUM OF DIGITS.cpp MAX ND MIN LIST.cpp TDM-GCC 4.9.2 64-bit Release
C:\Users\khato\OneDrive\Documents\DAACONTAINER LOADER.cpp x + - x CONTAINER LOADER.cpp
Enter the number of items: 3
Enter the container capacity: 3
Enter the weights of the items:
Item 1: 3
Item 2: 4
Item 3: 5
Item 1 of weight 3 placed in new container 1
Item 2 of weight 4 placed in new container 2
Item 3 of weight 5 placed in new container 3
Total containers used: 3
-----
Process exited after 3.994 seconds with return value 0
Press any key to continue . . .

```

The code in the editor is:

```

28     }
29
30     printf("\nTotal containers used: %d\n", containerCount);
31 }
32
33 int main() {
34     int items[MAX_ITEMS];
35     int n, containerCapacity;
36
37     // Input: number of items and container capacity
38     printf("Enter the number of items: ");
39     scanf("%d", &n);
40
41     printf("Enter the container capacity: ");
42     scanf("%d", &containerCapacity);
43
44     // Input: weight of each item
45     printf("Enter the weights of the items:\n");
46     for (int i = 0; i < n; i++) {
47         printf("Item %d: ", i+1);
48         scanf("%d", &items[i]);
49     }
50
51     // Call the function to find the container Loading solution
52     firstFit(items, n, containerCapacity);
53
54     return 0;
55 }

```

37. LIST OF ALL FACTOR

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

```

void findFactors(int n) {

    printf("Factors of %d are: ", n);
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            printf("%d ", i);
        }
    }
}

```

```
    }
    printf("\n");
}

int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    if (n <= 0) {
        printf("Please enter a positive integer.\n");
    } else {
        findFactors(n);
    }
}

return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ source code file named 'LIST OF ALL FACTOR.cpp'. The code implements a function to find factors of a given integer and prints them. A terminal window shows the output of running the program with input '4', displaying the factors '1 2 4'. Below the terminal is a status bar indicating the process exited after 1.641 seconds. The bottom panel shows the compiler's compilation results, which are successful with no errors or warnings.

```

1 #include <stdio.h>
2
3 void findFactors(int n) {
4     printf("Factors of %d are: ", n);
5     for (int i = 1; i <= n; i++) {
6         if (n % i == 0) {
7             printf("%d ", i);
8         }
9     }
10    printf("\n");
11}
12
13 int main() {
14     int n;
15     printf("Enter a positive integer: ");
16     scanf("%d", &n);
17
18     if (n <= 0) {
19         printf("Please enter a positive integer.\n");
20     } else {
21         findFactors(n);
22     }
23
24     return 0;
25}
26

```

38. BRANCH AND BOUND

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

```
#include <limits.h>
```

```
#include <stdbool.h>
```

```
#define N 4 // Number of cities
```

```
int costMatrix[N][N] = {
    {0, 10, 15, 20},
    {10, 0, 35, 25},
    {15, 35, 0, 30},
    {20, 25, 30, 0}
};
```

```
int finalRes = INT_MAX;
```

```

int finalPath[N + 1];

void TSP(int currPos, int n, int count, int cost, int path[]) {
    if (count == n && cost + costMatrix[currPos][0] < finalRes) {
        finalPath[count] = 0;
        finalRes = cost + costMatrix[currPos][0];
        return;
    }

    for (int i = 0; i < n; i++) {
        if (i != currPos && path[i] == 0) {
            int temp = cost + costMatrix[currPos][i];
            if (temp < finalRes) {
                path[i] = 1;
                TSP(i, n, count + 1, temp, path);
                path[i] = 0; // Backtrack
            }
        }
    }
}

int main() {
    int path[N] = {0}; // Initialize path array
    path[0] = 1; // Start from the first city
}

```

```
TSP(0, N, 1, 0, path); // Start TSP from city 0
```

```
printf("Minimum cost: %d\n", finalRes);
printf("Path: ");
for (int i = 0; i <= N; i++) {
    printf("%d ", finalPath[i]);
}
printf("\n");

return 0;
}
```

```
C:\Users\khato\OneDrive\Documents\DAAl\BRANCH ND BOUNCE.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug FLOYD.cpp PASCAL TRIANGLE.cpp OBSTUSING KNAPSACK.cpp SUM OF DIGITS.cpp MAX ND MIN LIST.cpp N QUEENS.cpp INSERT NUMBER.cpp LIST OF ALL FACTOR.cpp CONTAINER LOADER.cpp IDM-GCC 4.9.2 64-bit Release
C:\Users\khato\OneDrive\Doc x + -
Minimum cost: 80
Path: 0 0 0 0
-----
Process exited after 0.05165 seconds with return value 0
Press any key to continue . . .

```

```
24 for (int i = 0; i < n; i++) {
25     if (i != currPos && path[i] == 0) {
26         int temp = cost + costMatrix[currPos][i];
27         if (temp < finalRes) {
28             path[i] = 1;
29             TSP(i, n, count + 1, temp, path);
30             path[i] = 0; // Backtrack
31         }
32     }
33 }
34 }
35
36 int main() {
37     int path[N] = {0}; // Initialize path array
38     path[0] = 1; // Start from the first city
39
40     TSP(0, N, 1, 0, path); // Start TSP from city 0
41
42     printf("Minimum cost: %d\n", finalRes);
43     printf("Path: ");
44     for (int i = 0; i <= N; i++) {
45         printf("%d ", finalPath[i]);
46     }
47     printf("\n");
48
49     return 0;
50 }
```

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\khato\OneDrive\Documents\DAAl\BRANCH ND BOUNCE.
- Output Size: 129.2275390625 KIB
- Compilation Time: 0.22s

Line: 51 Col: 1 Sek: 0 Lines: 51 Length: 1204 Insert Done parsing in 0.016 seconds

39. LINEAR SEARCH

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

```
// Function for linear search
```

```
int linearSearch(int arr[], int size, int target) {  
    for (int i = 0; i < size; i++) {  
        if (arr[i] == target) {  
            return i; // Return the index if found  
        }  
    }  
    return -1; // Return -1 if not found  
}
```

```
int main() {  
    int arr[] = {2, 4, 0, 1, 9, 6};  
    int size = sizeof(arr) / sizeof(arr[0]);  
    int target;  
  
    printf("Enter the number to search: ");  
    scanf("%d", &target);  
  
    int result = linearSearch(arr, size, target);  
  
    if (result != -1) {  
        printf("Element found at index: %d\n", result);  
    } else {  
        printf("Element not found in the array.\n");  
    }  
}
```

```

        return 0;
    }
}

```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C++ file named LINEAR SEARCH.cpp. The code implements a linear search algorithm. The main function initializes an array with values {2, 4, 0, 1, 9, 6}, sets its size to 6, and prompts the user to enter a target value. It then calls the linearSearch function and prints the result. The output window shows the user entering '4' as the target, the program finding it at index 1, and exiting after 1.44 seconds. The status bar at the bottom shows system information like battery level (94%), date (9/23/2024), and time (12:34 PM).

```

C:\Users\khato\OneDrive\Documents\DAALINEAR SEARCH.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASStyle Window Help
Project Classes Debug
FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNPACK.cpp
GRAPH COLORING.cpp CONTAINER LOADER.cpp
4 int linearSearch(int arr[], int size, int target) {
5     for (int i = 0; i < size; i++) {
6         if (arr[i] == target) {
7             return i; // Return the index if found
8         }
9     }
10    return -1; // Return -1 if not found
11 }
12
13 int main() {
14     int arr[] = {2, 4, 0, 1, 9, 6};
15     int size = sizeof(arr) / sizeof(arr[0]);
16     int target;
17
18     printf("Enter the number to search: ");
19     scanf("%d", &target);
20
21     int result = linearSearch(arr, size, target);
22
23     if (result != -1) {
24         printf("Element found at index: %d\n", result);
25     } else {
26         printf("Element not found in the array.\n");
27     }
28
29     return 0;
30 }
31
Compiler Resources Compile Log Debug Find Results Close
Compilation results...
Abort Compilation
-----  

- Errors: 0  

- Warnings: 0  

- Output Filename: C:\Users\khato\OneDrive\Documents\DAALINEAR SEARCH.exe  

- Output Size: 129.30859375 Kib  

- Compilation Time: 0.27s
94% Partly sunny
ENG IN 12:34 PM 9/23/2024

```

40. HAMILTONIAN

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

```
#include <stdbool.h>
```

```
#define V 5 // Number of vertices in the graph
```

```
bool isSafe(int graph[V][V], int path[], int pos, int v) {
```

```
    // Check if this vertex is an adjacent vertex of the previously added vertex.
```

```
    if (graph[path[pos - 1]][v] == 0) {
```

```
        return false;
```

```
}
```

```
// Check if the vertex has already been included in the path.
```

```

for (int i = 0; i < pos; i++) {
    if (path[i] == v) {
        return false;
    }
}

return true;
}

```

```

bool hamCycleUtil(int graph[V][V], int path[], int pos) {
    // Base case: If all vertices are included in the path
    if (pos == V) {
        // And if there is an edge from the last included vertex to the first vertex
        return graph[path[pos - 1]][path[0]] == 1;
    }

    // Try different vertices as the next candidate in the Hamiltonian Path
    for (int v = 1; v < V; v++) {
        if (isSafe(graph, path, pos, v)) {
            path[pos] = v;

            if (hamCycleUtil(graph, path, pos + 1)) {
                return true;
            }
        }
    }
}

```

```

// Backtrack
path[pos] = -1;
}

}

return false;
}

void findHamiltonianCycle(int graph[V][V]) {
    int path[V];
    for (int i = 0; i < V; i++) {
        path[i] = -1;
    }

    // Start from the first vertex
    path[0] = 0;

    if (!hamCycleUtil(graph, path, 1)) {
        printf("No Hamiltonian Cycle found\n");
    } else {
        printf("Hamiltonian Cycle found:\n");
        for (int i = 0; i < V; i++) {
            printf("%d ", path[i]);
        }
        printf("%d\n", path[0]); // to show the cycle
    }
}

```

```
    }  
}  
  
}
```

```
int main() {  
    // Example graph (adjacency matrix representation)  
    int graph[V][V] = {  
        {0, 1, 0, 1, 0},  
        {1, 0, 1, 0, 1},  
        {0, 1, 0, 1, 0},  
        {1, 0, 1, 0, 1},  
        {0, 1, 0, 1, 0}  
    };  
  
    findHamiltonianCycle(graph);  
  
    return 0;  
}
```

C:\Users\khato\OneDrive\Documents\DAAl HAMILTONIAN.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

Project Classes Debug FLOYD.cpp PASCAL TRIANGLE.cpp OBST USING KNAPSACK.cpp LIST OF FILES (globals)

GRAPH COLORING.cpp CONTAINER LOADER.cpp

53 path[0] = 0;

54 if (!hamCycleUtil(graph, path, 1)) {

55 printf("No Hamiltonian Cycle found\n");

56 } else {

57 printf("Hamiltonian Cycle found:\n");

58 for (int i = 0; i < V; i++) {

59 | printf("%d ", path[i]);

60 }

61 }

62 printf("\n", path[0]); // to show the cycle

63 }

64 }

65 }

66 int main() {

67 // Example graph (adjacency matrix representation)

68 int graph[V][V] = {

69 {0, 1, 0, 1, 0},

70 {1, 0, 1, 0, 1},

71 {0, 1, 0, 1, 0},

72 {1, 0, 1, 0, 1},

73 {0, 1, 0, 1, 0}

74 };

75

76 findHamiltonianCycle(graph);

77

78 return 0;

79 }

80

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0

- Warnings: 0

- Output Filename: C:\Users\khato\OneDrive\Documents\DAAl HAMILTONIAN.exe

- Output Size: 129.2255859375 Kib

- Compilation Time: 0.22s

Line: 80 Col: 1 Sel: 0 Lines: 80 Length: 1970 Insert Done parsing in 0.016 seconds

No Hamiltonian Cycle found

Process exited after 0.05109 seconds with return value 0

Press any key to continue . . . |