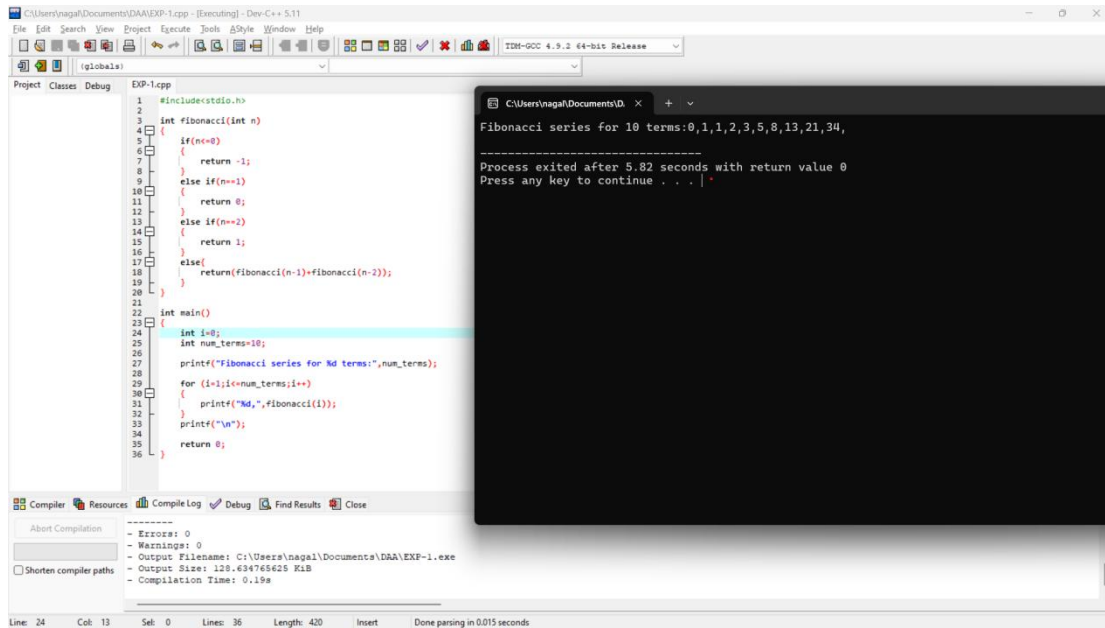


40 PROGRAMS

JEEVAN GOLI (192211714)

CSA0659-DESIGN AND ANALYSIS OF ALGORITHMS FOR NETWORK PROTOCOLS

1.



The screenshot shows a C++ IDE with a file named EXP-1.cpp. The code implements a recursive Fibonacci function and a main function that prints the first 10 terms of the series. The output window displays the series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34. The process exited after 5.82 seconds.

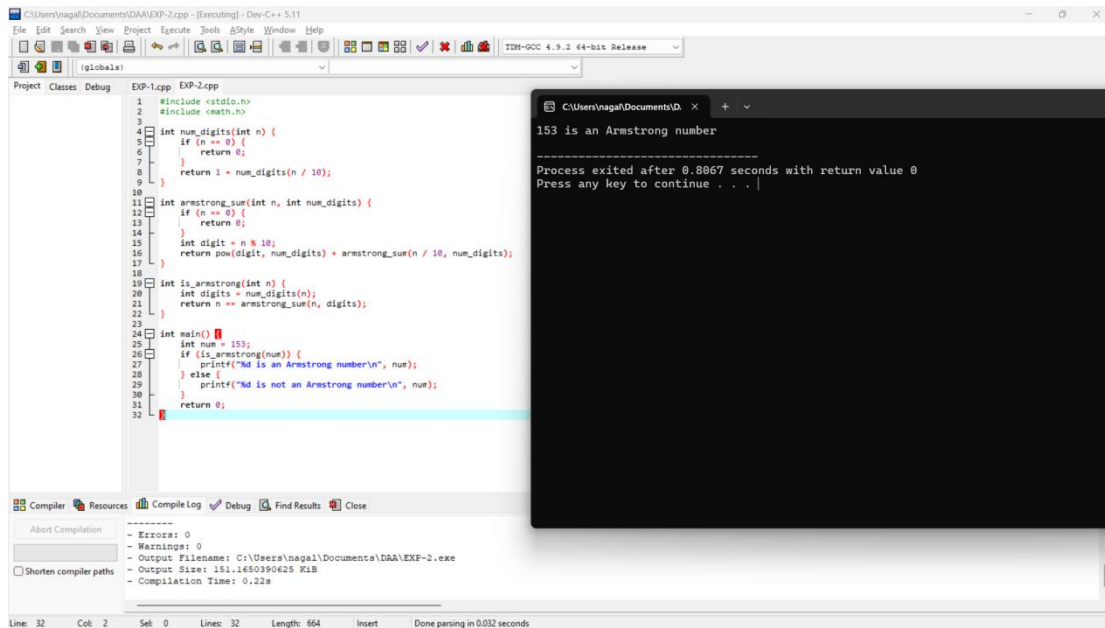
```
#include <stdio.h>

int fibonacci(int n)
{
    if(n==0)
        return -1;
    else if(n==1)
        return 0;
    else if(n==2)
        return 1;
    else
        return (fibonacci(n-1)+fibonacci(n-2));
}

int main()
{
    int i=0;
    int num_terms=10;
    printf("Fibonacci series for %d terms:", num_terms);
    for (i=1; i<=num_terms; i++)
    {
        printf("%d, ", fibonacci(i));
        printf("\n");
    }
    return 0;
}
```

Output: Fibonacci series for 10 terms:0,1,1,2,3,5,8,13,21,34,
Process exited after 5.82 seconds with return value 0
Press any key to continue . . .

2.



The screenshot shows a C++ IDE with a file named EXP-2.cpp. The code implements functions to count digits, calculate the sum of their cubes, and check if a number is an Armstrong number. The main function checks the number 153. The output window displays: 153 is an Armstrong number. The process exited after 0.8067 seconds.

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

int num_digits(int n) {
    if (n == 0) {
        return 0;
    }
    return 1 + num_digits(n / 10);
}

int armstrong_sum(int n, int num_digits) {
    if (n == 0) {
        return 0;
    }
    int digit = n % 10;
    return pow(digit, num_digits) + armstrong_sum(n / 10, num_digits);
}

int is_armstrong(int n) {
    int digits = num_digits(n);
    return n == armstrong_sum(n, digits);
}

int main() {
    int num = 153;
    if (is_armstrong(num)) {
        printf("153 is an Armstrong number\n");
    } else {
        printf("153 is not an Armstrong number\n");
    }
    return 0;
}
```

Output: 153 is an Armstrong number
Process exited after 0.8067 seconds with return value 0
Press any key to continue . . .

3.

The screenshot shows a C++ IDE with the following code in `EXP-3.cpp`:

```

1 #include<stdio.h>
2
3 int gcd(int a,int b)
4 {
5     if(b==0){
6         return a;
7     }
8     else{
9         return(b,a%b);
10    }
11 }
12
13 int main()
14 {
15     int num1=20;
16     int num2=40;
17
18     printf("GCD of %d & %d is %d",num1,num2,gcd(num1,num2));
19
20     return 0;
21 }

```

The output window displays:

```

GCD of 20 & 40 is 20
-----
Process exited after 2.016 seconds with return value 0
Press any key to continue . . .

```

The compiler log at the bottom shows no errors or warnings.

4.

The screenshot shows a C++ IDE with the following code in `EXP-4.cpp`:

```

1 #include <stdio.h>
2
3 int find_largest(int arr[], int n) {
4     if (n == 1) {
5         return arr[0];
6     }
7
8     int max_of_rest = find_largest(arr, n - 1);
9     return (arr[n - 1] > max_of_rest) ? arr[n - 1] : max_of_rest;
10 }
11
12 int main() {
13     int arr[] = {3, 5, 7, 2, 8, 1};
14     int n = sizeof(arr) / sizeof(arr[0]);
15     printf("The largest element in the array is %d\n", find_largest(arr, n));
16
17     return 0;
18 }

```

The output window displays:

```

The largest element in the array is 8
-----
Process exited after 1.81 seconds with return value 0
Press any key to continue . . .

```

The compiler log at the bottom shows no errors or warnings.

5.

C:\Users\nagaj\Documents\DA\EXP-5.cpp - [Executing] - Dev-C++ 5.11

```

1 #include<stdio.h>
2
3 int factorial(int n){
4     if(n==0||n==1){
5         return 1;
6     }
7     else{
8         return n*factorial(n-1);
9     }
10 }
11
12 int main(){
13     int num=5;
14
15     printf("Factorial of %d is %d",num,factorial(num));
16
17     return 0;
18 }

```

Compiler Resources Compile Log Debug Find Results Close

 - Errors: 0
 - Warnings: 0
 - Output Filename: C:\Users\nagaj\Documents\DA\EXP-5.exe
 - Output Size: 128.46289625 KiB
 - Compilation Time: 0.10s

Line: 17 Col: 14 Sel: 0 Lines: 18 Length: 225 Insert Done parsing in 0 seconds

C:\Users\nagaj\Documents\DA

```

Factorial of 5 is 120
-----
Process exited after 0.6644 seconds with return value 0
Press any key to continue . . .

```

6.

C:\Users\nagaj\Documents\DA\EXP-6.cpp - [Executing] - Dev-C++ 5.11

```

1 #include<stdio.h>
2 int main(){
3     int i,n,flag=0;
4     printf("Enter the number:");
5     scanf("%d",&n);
6     if (n==0||n==1){
7         flag=1;
8     }
9
10    for(i=2;i<=n/2;i++){
11        if(n%i==0){
12            flag=1;
13            break;
14        }
15    }
16
17    if(flag==0){
18        printf(" %d is a prime number...",n);
19    }
20    else{
21        printf(" %d is not a prime number...",n);
22    }
23
24    return 0;
25 }

```

Compiler Resources Compile Log Debug Find Results Close

 - Errors: 0
 - Warnings: 0
 - Output Filename: C:\Users\nagaj\Documents\DA\EXP-6.exe
 - Output Size: 128.6015625 KiB
 - Compilation Time: 0.20s

Line: 24 Col: 12 Sel: 0 Lines: 25 Length: 342 Insert Done parsing in 0 seconds

C:\Users\nagaj\Documents\DA

```

Enter the number:2
2 is a prime number...
-----
Process exited after 7.306 seconds with return value 0
Press any key to continue . . .

```

7.

The screenshot shows a C++ IDE with a project named 'EXP-7.cpp'. The code implements a selection sort algorithm. The array to be sorted is {64, 25, 12, 22, 11}. The output shows the sorted array: 11 12 22 25 64. The process exited after 0.7123 seconds with a return value of 0.

```

1 #include <stdio.h>
2
3 void selection_sort(int arr[], int n) {
4     for (int i = 0; i < n-1; i++) {
5         int min_idx = i;
6         for (int j = i+1; j < n; j++) {
7             if (arr[j] < arr[min_idx]) {
8                 min_idx = j;
9             }
10        }
11        int temp = arr[min_idx];
12        arr[min_idx] = arr[i];
13        arr[i] = temp;
14    }
15 }
16
17
18 int main() {
19     int arr[] = {64, 25, 12, 22, 11};
20     int n = sizeof(arr)/sizeof(arr[0]);
21     selection_sort(arr, n);
22     printf("Sorted array: ");
23     for (int i = 0; i < n; i++) {
24         printf("%d ", arr[i]);
25     }
26     printf("\n");
27     return 0;
28 }

```

Sorted array: 11 12 22 25 64

Process exited after 0.7123 seconds with return value 0
Press any key to continue . . .

Compiler: GCC 4.9.2 64-bit Release
Output Filename: C:\Users\nagai\Documents\DA\EXP-7.exe
Output Size: 128.642878125 KiB
Compilation Time: 0.25s

8.

The screenshot shows a C++ IDE with a project named 'EXP-8.cpp'. The code implements a bubble sort algorithm. The array to be sorted is {64, 34, 25, 12, 22, 11, 90}. The output shows the sorted array: 11 12 22 25 34 64 90. The process exited after 4.583 seconds with a return value of 0.

```

1 #include <stdio.h>
2
3 void bubble_sort(int arr[], int n) {
4     for (int i = 0; i < n-1; i++) {
5         for (int j = 0; j < n-i-1; j++) {
6             if (arr[j] > arr[j+1]) {
7                 int temp = arr[j];
8                 arr[j] = arr[j+1];
9                 arr[j+1] = temp;
10            }
11        }
12    }
13 }
14
15 int main() {
16     int arr[] = {64, 34, 25, 12, 22, 11, 90};
17     int n = sizeof(arr)/sizeof(arr[0]);
18     bubble_sort(arr, n);
19     printf("Sorted array: ");
20     for (int i = 0; i < n; i++) {
21         printf("%d ", arr[i]);
22     }
23     printf("\n");
24     return 0;
25 }

```

Sorted array: 11 12 22 25 34 64 90

Process exited after 4.583 seconds with return value 0
Press any key to continue . . .

Compiler: GCC 4.9.2 64-bit Release
Output Filename: C:\Users\nagai\Documents\DA\EXP-8.exe
Output Size: 128.6396484375 KiB
Compilation Time: 0.17s

9.

```

1 #include <iostream.h>
2 using namespace std;
3
4 void inputMatrix(int matrix[][10], int rows, int cols) {
5     for (int i = 0; i < rows; i++)
6         for (int j = 0; j < cols; j++)
7             cin >> matrix[i][j];
8 }
9
10 void printMatrix(int matrix[][10], int rows, int cols) {
11     for (int i = 0; i < rows; i++)
12         for (int j = 0; j < cols; j++)
13             cout << matrix[i][j] << " ";
14     cout << endl;
15 }
16
17 void multiplyMatrix(int matrix1[][10], int matrix2[][10], int result[][10], int rows1, int cols1, int cols2) {
18     for (int i = 0; i < rows1; i++)
19         for (int j = 0; j < cols2; j++)
20             result[i][j] = 0;
21     for (int i = 0; i < rows1; i++)
22         for (int j = 0; j < cols2; j++)
23             result[i][j] = matrix1[i][0] * matrix2[0][j];
24 }
25
26 int main() {
27     int matrix1[10][10], matrix2[10][10], result[10][10];
28     int rows1, cols1, rows2, cols2;
29
30     printMatrix(matrix1, rows1, cols1);
31     printMatrix(matrix2, rows2, cols2);
32
33     multiplyMatrix(matrix1, matrix2, result, rows1, cols1, cols2);
34     printMatrix(result, rows1, cols1);
35
36     return 0;
37 }

```

Enter rows and columns for the first matrix: 2 2
Enter rows and columns for the second matrix: 2 2
Enter the first matrix: 1 2 3 4
Enter the second matrix: 5 6 7 8
Resultant matrix:
19 22
43 50

Process exited after 12.01 seconds with return value 0
Press any key to continue . . .

10.

```

1 #include <iostream.h>
2 #include <string.h>
3 #include <string.h>
4
5 bool is_palindrome(char *s, int start, int end) {
6     if (start >= end)
7         return true;
8     if (s[start] == s[end])
9         return is_palindrome(s, start + 1, end - 1);
10    else
11        return false;
12 }
13
14 int main() {
15     char string[] = "racecar";
16     int length = strlen(string);
17     if (is_palindrome(string, 0, length - 1)) {
18         printf("%s is a palindrome\n", string);
19     } else {
20         printf("%s is not a palindrome\n", string);
21     }
22     return 0;
23 }

```

racecar is a palindrome

Process exited after 2.806 seconds with return value 0
Press any key to continue . . .

11.

Dev-C++ IDE showing the execution of a C++ program for string copying. The code defines a `copyString` function and a `main` function that prompts the user for a string to copy and a source string. The output window shows the input 'sai' and the copied string 'sai'.

```

1 #include <stdio.h>
2
3 void copyString(char source[], char destination[], int index) {
4     if (source[index] == '\0') {
5         destination[index] = '\0';
6         return;
7     }
8     destination[index] = source[index];
9     copyString(source, destination, index + 1);
10 }
11
12 int main() {
13     char source[100], destination[100];
14
15     printf("Enter a string to copy: ");
16     fgets(source, sizeof(source), stdin);
17     copyString(source, destination, 0);
18
19     printf("Source string: %s\n", source);
20     printf("Copied string: %s\n", destination);
21
22     return 0;
23 }

```

Output window:

```

Enter a string to copy: sai
Source string: sai
Copied string: sai

-----
Process exited after 6.487 seconds with return value 0
Press any key to continue . . .

```

12.

Dev-C++ IDE showing the execution of a C++ program for binary search. The code defines a `binarySearch` function and a `main` function that prompts the user for a key to search in an array. The output window shows the key '12' and the result 'Element found at index 5'.

```

1 #include <stdio.h>
2
3 int binarySearch(int arr[], int left, int right, int key) {
4     if (right >= left) {
5         int mid = left + (right - left) / 2;
6
7         if (arr[mid] == key)
8             return mid;
9
10        if (arr[mid] > key)
11            return binarySearch(arr, left, mid - 1, key);
12        return binarySearch(arr, mid + 1, right, key);
13    }
14    return -1;
15 }
16
17 int main() {
18     int arr[] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
19     int n = sizeof(arr) / sizeof(arr[0]);
20     int key = 12;
21     int result = binarySearch(arr, 0, n - 1, key);
22
23     if (result == -1)
24         printf("Element not found.\n");
25     else
26         printf("Element found at index %d.\n", result);
27
28     return 0;
29 }

```

Output window:

```

Element found at index 5.

-----
Process exited after 1.731 seconds with return value 0
Press any key to continue . . .

```

13.

```

1 #include <stdio.h>
2 #include <string.h>
3
4 void reverseString(char str[], int start, int end) {
5     if (start >= end) {
6         return;
7     }
8     char temp = str[start];
9     str[start] = str[end];
10    str[end] = temp;
11    reverseString(str, start + 1, end - 1);
12 }
13
14 int main() {
15     char str[100];
16     printf("Enter a string: ");
17     fgets(str, sizeof(str), stdin);
18     str[strlen(str) - 1] = '\0';
19     reverseString(str, 0, strlen(str) - 1);
20     printf("Reversed string: %s\n", str);
21     return 0;
22 }

```

Enter a string: sai
Reversed string: ias

Process exited after 7.174 seconds with return value 0
Press any key to continue . . .

14.

```

1 #include <stdio.h>
2
3 int main() {
4     int numbers[] = {4, 7, 2, 9, 1, 5, 8, 3, 6};
5     int n = sizeof(numbers) / sizeof(numbers[0]);
6
7     int minSequence = numbers[0];
8     int maxSequence = numbers[0];
9
10    for (int i = 1; i < n; i++) {
11        if (numbers[i] < minSequence) {
12            minSequence = numbers[i];
13        }
14        if (numbers[i] > maxSequence) {
15            maxSequence = numbers[i];
16        }
17    }
18
19    printf("Minimum value sequence: %d\n", minSequence);
20    printf("Maximum value sequence: %d\n", maxSequence);
21    return 0;
22 }

```

Minimum value sequence: 1
Maximum value sequence: 9

Process exited after 0.9998 seconds with return value 0
Press any key to continue . . .

15.

The screenshot shows a C++ IDE with a project named 'EXP-15.cpp'. The code implements Strassen's Matrix Multiplication for two 3x3 matrices A and B, resulting in matrix C. The output window displays the resultant matrix C as:

```
Resultant Matrix after Strassen's Matrix Multiplication:
19 22
43 50
```

The process exited after 1.369 seconds with return value 0. The code in the editor is as follows:

```
#include <stdio.h>

void strassenMatrixMultiply(int A[3][3], int B[3][3], int C[3][3]) {
    int M1 = (A[0][0] + A[1][1]) * (B[0][0] + B[1][1]);
    int M2 = (A[1][0] + A[2][1]) * B[0][0];
    int M3 = A[0][0] * (B[0][1] + B[1][1]);
    int M4 = A[1][1] * (B[1][0] - B[0][0]);
    int M5 = (A[0][0] + A[0][1]) * B[1][1];
    int M6 = (A[1][0] - A[0][0]) * (B[0][1] + B[1][1]);
    int M7 = (A[0][1] - A[1][1]) * (B[1][0] + B[1][1]);

    C[0][0] = M1 + M4 - M5 + M7;
    C[0][1] = M3 + M5;
    C[1][0] = M2 + M4;
    C[1][1] = M1 - M2 + M3 + M6;
}

int main() {
    int A[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int B[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int C[3][3];

    strassenMatrixMultiply(A, B, C);

    printf("Resultant Matrix after Strassen's Matrix Multiplication:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

16.

The screenshot shows a C++ IDE with a project named 'EXP-16.cpp'. The code implements Merge Sort on an array of numbers. The output window displays the sorted array as:

```
Sorted array: 5 6 7 11 12 13
```

The process exited after 1.086 seconds with return value 0. The code in the editor is as follows:

```
#include <iostream>

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int *L = new int[n1];
    int *R = new int[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;

    while (i < n1 && j < n2) {
        if (L[i] < R[j])
            arr[k++] = L[i++];
        else
            arr[k++] = R[j++];
    }

    while (i < n1)
        arr[k++] = L[i++];
    while (j < n2)
        arr[k++] = R[j++];

    delete[] L;
    delete[] R;
}

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

    mergeSort(arr, 0, n - 1);

    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```


17.

```

1 #include <stdio.h>
2 void findMaxMin(int arr[], int low, int high, int *max, int *min) {
3     int mid, max1, max2, min1, min2;
4     if (low == high) {
5         *max = arr[low];
6         *min = arr[low];
7     }
8     else if (high - low == 1) {
9         if (arr[low] > arr[high]) {
10             *max = arr[low];
11             *min = arr[high];
12         } else {
13             *max = arr[high];
14             *min = arr[low];
15         }
16     }
17     else {
18         mid = (low + high) / 2;
19         findMaxMin(arr, low, mid, &max1, &min1);
20         findMaxMin(arr, mid + 1, high, &max2, &min2);
21         *max = (max1 > max2) ? max1 : max2;
22         *min = (min1 < min2) ? min1 : min2;
23     }
24 }
25
26 int main() {
27     int arr[] = {7, 3, 9, 1, 5, 12, 6};
28     int n = sizeof(arr) / sizeof(arr[0]);
29     int max, min;
30
31     findMaxMin(arr, 0, n - 1, &max, &min);
32     printf("Maximum value: %d\n", max);
33     printf("Minimum value: %d\n", min);
34
35     return 0;
36 }

```

Maximum value: 12
Minimum value: 1

Process exited after 0.9031 seconds with return value 0
Press any key to continue . . .

18.

```

1 #include <stdio.h>
2 #include <stdbool.h>
3 bool isPrime(int num, int i) {
4     if (i == 1) {
5         return true;
6     }
7     else {
8         if (num % i == 0) {
9             return false;
10        } else {
11            return isPrime(num, i + 1);
12        }
13    }
14 }
15
16 void generatePrimes(int n, int i) {
17     if (i == n) {
18         return;
19     }
20     if (isPrime(i, 1)) {
21         printf("%d is a prime number\n", i);
22     }
23     generatePrimes(n, i + 1);
24 }
25
26 int main() {
27     int n;
28
29     printf("Enter the value of n: ");
30     scanf("%d", &n);
31
32     printf("Prime numbers up to %d are:\n", n);
33     generatePrimes(n, 2);
34
35     return 0;
36 }

```

Enter the value of n: 10
Prime numbers up to 10 are:
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number

Process exited after 11.35 seconds with return value 0
Press any key to continue . . .

19.

The screenshot shows a C++ IDE with a project named 'EXP-19.cpp'. The code implements a greedy algorithm for the knapsack problem. It defines a struct 'Item' with 'value' and 'weight' attributes. The 'knapsackGreedy' function iterates through items, sorting them by value/weight ratio, and selects items until the knapsack is full. The 'main' function initializes a knapsack with capacity 50 and a set of items, then calls the greedy function.

```

4 // int value;
5 //
6 // void knapsackGreedy(int W, struct Item items[], int n) {
7 //     int i, j, currentWeight;
8 //     float totalValue = 0.0;
9 //     for (i = 0; i < n; i++) {
10 //         items[i].value = items[i].value / items[i].weight;
11 //     }
12 //     for (i = 0; i < n; i++) {
13 //         for (j = i + 1; j < n; j++) {
14 //             if (items[i].value < items[j].value) {
15 //                 struct Item temp = items[i];
16 //                 items[i] = items[j];
17 //                 items[j] = temp;
18 //             }
19 //         }
20 //     }
21 //     for (i = 0; i < n; i++) {
22 //         if (items[i].weight <= W) {
23 //             totalValue += items[i].value;
24 //             W -= items[i].weight;
25 //         } else {
26 //             totalValue += items[i].value * ((float) W / items[i].weight);
27 //             break;
28 //         }
29 //     }
30 //     printf("Maximum value in Knapsack: %.2f\n", totalValue);
31 // }
32 //
33 // int main() {
34 //     int W = 50;
35 //     struct Item items[] = {{10, 60}, {20, 100}, {30, 120}};
36 //     int n = sizeof(items) / sizeof(items[0]);
37 //     knapsackGreedy(W, items, n);
38 //     return 0;
39 // }
40 //
41 //
42 //

```

The output window shows the result of the execution:

```

C:\Users\nagaj\Documents\EXP-19.exe
Maximum value in Knapsack: 13.67
Process exited after 1.161 seconds with return value 0
Press any key to continue . . .

```

Compiler output: 0 Errors, 0 Warnings. Output File: C:\Users\nagaj\Documents\EXP-19.exe. Output Size: 129.975589975 KiB. Compilation Time: 0.39s.

20.

The screenshot shows a C++ IDE with a project named 'EXP-20.cpp'. The code implements a graph algorithm, likely for finding a Minimum Spanning Tree (MST) using Kruskal's algorithm. It defines a struct 'Edge' with 'source', 'target', and 'weight' attributes. The 'kruskal' function sorts edges by weight and adds them to the MST if they do not create a cycle. The 'main' function initializes a graph with 5 nodes and 6 edges, then calls the kruskal function.

```

4 // struct Edge {
5 //     int source, target, weight;
6 // };
7 //
8 // void kruskal(struct Edge edges[], int n) {
9 //     int i, j, k;
10 //     sort(edges, edges + n, [](const Edge& a, const Edge& b) { return a.weight < b.weight; });
11 //     for (i = 0; i < n; i++) {
12 //         for (j = i + 1; j < n; j++) {
13 //             if (edges[i].weight < edges[j].weight) {
14 //                 swap(edges[i], edges[j]);
15 //             }
16 //         }
17 //     }
18 //     for (i = 0; i < n; i++) {
19 //         for (j = i + 1; j < n; j++) {
20 //             if (edges[i].weight < edges[j].weight) {
21 //                 swap(edges[i], edges[j]);
22 //             }
23 //         }
24 //     }
25 //     for (i = 0; i < n; i++) {
26 //         for (j = i + 1; j < n; j++) {
27 //             if (edges[i].weight < edges[j].weight) {
28 //                 swap(edges[i], edges[j]);
29 //             }
30 //         }
31 //     }
32 //     for (i = 0; i < n; i++) {
33 //         for (j = i + 1; j < n; j++) {
34 //             if (edges[i].weight < edges[j].weight) {
35 //                 swap(edges[i], edges[j]);
36 //             }
37 //         }
38 //     }
39 //     for (i = 0; i < n; i++) {
40 //         for (j = i + 1; j < n; j++) {
41 //             if (edges[i].weight < edges[j].weight) {
42 //                 swap(edges[i], edges[j]);
43 //             }
44 //         }
45 //     }
46 //     for (i = 0; i < n; i++) {
47 //         for (j = i + 1; j < n; j++) {
48 //             if (edges[i].weight < edges[j].weight) {
49 //                 swap(edges[i], edges[j]);
50 //             }
51 //         }
52 //     }
53 //     for (i = 0; i < n; i++) {
54 //         for (j = i + 1; j < n; j++) {
55 //             if (edges[i].weight < edges[j].weight) {
56 //                 swap(edges[i], edges[j]);
57 //             }
58 //         }
59 //     }
60 //     for (i = 0; i < n; i++) {
61 //         for (j = i + 1; j < n; j++) {
62 //             if (edges[i].weight < edges[j].weight) {
63 //                 swap(edges[i], edges[j]);
64 //             }
65 //         }
66 //     }
67 //     for (i = 0; i < n; i++) {
68 //         for (j = i + 1; j < n; j++) {
69 //             if (edges[i].weight < edges[j].weight) {
70 //                 swap(edges[i], edges[j]);
71 //             }
72 //         }
73 //     }
74 //     for (i = 0; i < n; i++) {
75 //         for (j = i + 1; j < n; j++) {
76 //             if (edges[i].weight < edges[j].weight) {
77 //                 swap(edges[i], edges[j]);
78 //             }
79 //         }
80 //     }
81 //     for (i = 0; i < n; i++) {
82 //         for (j = i + 1; j < n; j++) {
83 //             if (edges[i].weight < edges[j].weight) {
84 //                 swap(edges[i], edges[j]);
85 //             }
86 //         }
87 //     }
88 //     for (i = 0; i < n; i++) {
89 //         for (j = i + 1; j < n; j++) {
90 //             if (edges[i].weight < edges[j].weight) {
91 //                 swap(edges[i], edges[j]);
92 //             }
93 //         }
94 //     }
95 //     for (i = 0; i < n; i++) {
96 //         for (j = i + 1; j < n; j++) {
97 //             if (edges[i].weight < edges[j].weight) {
98 //                 swap(edges[i], edges[j]);
99 //             }
100 //         }
101 //     }
102 //     for (i = 0; i < n; i++) {
103 //         for (j = i + 1; j < n; j++) {
104 //             if (edges[i].weight < edges[j].weight) {
105 //                 swap(edges[i], edges[j]);
106 //             }
107 //         }
108 //     }
109 //     for (i = 0; i < n; i++) {
110 //         for (j = i + 1; j < n; j++) {
111 //             if (edges[i].weight < edges[j].weight) {
112 //                 swap(edges[i], edges[j]);
113 //             }
114 //         }
115 //     }
116 //     for (i = 0; i < n; i++) {
117 //         for (j = i + 1; j < n; j++) {
118 //             if (edges[i].weight < edges[j].weight) {
119 //                 swap(edges[i], edges[j]);
120 //             }
121 //         }
122 //     }
123 //     for (i = 0; i < n; i++) {
124 //         for (j = i + 1; j < n; j++) {
125 //             if (edges[i].weight < edges[j].weight) {
126 //                 swap(edges[i], edges[j]);
127 //             }
128 //         }
129 //     }
130 //     for (i = 0; i < n; i++) {
131 //         for (j = i + 1; j < n; j++) {
132 //             if (edges[i].weight < edges[j].weight) {
133 //                 swap(edges[i], edges[j]);
134 //             }
135 //         }
136 //     }
137 //     for (i = 0; i < n; i++) {
138 //         for (j = i + 1; j < n; j++) {
139 //             if (edges[i].weight < edges[j].weight) {
140 //                 swap(edges[i], edges[j]);
141 //             }
142 //         }
143 //     }
144 //     for (i = 0; i < n; i++) {
145 //         for (j = i + 1; j < n; j++) {
146 //             if (edges[i].weight < edges[j].weight) {
147 //                 swap(edges[i], edges[j]);
148 //             }
149 //         }
150 //     }
151 //     for (i = 0; i < n; i++) {
152 //         for (j = i + 1; j < n; j++) {
153 //             if (edges[i].weight < edges[j].weight) {
154 //                 swap(edges[i], edges[j]);
155 //             }
156 //         }
157 //     }
158 //     for (i = 0; i < n; i++) {
159 //         for (j = i + 1; j < n; j++) {
160 //             if (edges[i].weight < edges[j].weight) {
161 //                 swap(edges[i], edges[j]);
162 //             }
163 //         }
164 //     }
165 //     for (i = 0; i < n; i++) {
166 //         for (j = i + 1; j < n; j++) {
167 //             if (edges[i].weight < edges[j].weight) {
168 //                 swap(edges[i], edges[j]);
169 //             }
170 //         }
171 //     }
172 //     for (i = 0; i < n; i++) {
173 //         for (j = i + 1; j < n; j++) {
174 //             if (edges[i].weight < edges[j].weight) {
175 //                 swap(edges[i], edges[j]);
176 //             }
177 //         }
178 //     }
179 //     for (i = 0; i < n; i++) {
180 //         for (j = i + 1; j < n; j++) {
181 //             if (edges[i].weight < edges[j].weight) {
182 //                 swap(edges[i], edges[j]);
183 //             }
184 //         }
185 //     }
186 //     for (i = 0; i < n; i++) {
187 //         for (j = i + 1; j < n; j++) {
188 //             if (edges[i].weight < edges[j].weight) {
189 //                 swap(edges[i], edges[j]);
190 //             }
191 //         }
192 //     }
193 //     for (i = 0; i < n; i++) {
194 //         for (j = i + 1; j < n; j++) {
195 //             if (edges[i].weight < edges[j].weight) {
196 //                 swap(edges[i], edges[j]);
197 //             }
198 //         }
199 //     }
200 //     for (i = 0; i < n; i++) {
201 //         for (j = i + 1; j < n; j++) {
202 //             if (edges[i].weight < edges[j].weight) {
203 //                 swap(edges[i], edges[j]);
204 //             }
205 //         }
206 //     }
207 //     for (i = 0; i < n; i++) {
208 //         for (j = i + 1; j < n; j++) {
209 //             if (edges[i].weight < edges[j].weight) {
210 //                 swap(edges[i], edges[j]);
211 //             }
212 //         }
213 //     }
214 //     for (i = 0; i < n; i++) {
215 //         for (j = i + 1; j < n; j++) {
216 //             if (edges[i].weight < edges[j].weight) {
217 //                 swap(edges[i], edges[j]);
218 //             }
219 //         }
220 //     }
221 //     for (i = 0; i < n; i++) {
222 //         for (j = i + 1; j < n; j++) {
223 //             if (edges[i].weight < edges[j].weight) {
224 //                 swap(edges[i], edges[j]);
225 //             }
226 //         }
227 //     }
228 //     for (i = 0; i < n; i++) {
229 //         for (j = i + 1; j < n; j++) {
230 //             if (edges[i].weight < edges[j].weight) {
231 //                 swap(edges[i], edges[j]);
232 //             }
233 //         }
234 //     }
235 //     for (i = 0; i < n; i++) {
236 //         for (j = i + 1; j < n; j++) {
237 //             if (edges[i].weight < edges[j].weight) {
238 //                 swap(edges[i], edges[j]);
239 //             }
240 //         }
241 //     }
242 //     for (i = 0; i < n; i++) {
243 //         for (j = i + 1; j < n; j++) {
244 //             if (edges[i].weight < edges[j].weight) {
245 //                 swap(edges[i], edges[j]);
246 //             }
247 //         }
248 //     }
249 //     for (i = 0; i < n; i++) {
250 //         for (j = i + 1; j < n; j++) {
251 //             if (edges[i].weight < edges[j].weight) {
252 //                 swap(edges[i], edges[j]);
253 //             }
254 //         }
255 //     }
256 //     for (i = 0; i < n; i++) {
257 //         for (j = i + 1; j < n; j++) {
258 //             if (edges[i].weight < edges[j].weight) {
259 //                 swap(edges[i], edges[j]);
260 //             }
261 //         }
262 //     }
263 //     for (i = 0; i < n; i++) {
264 //         for (j = i + 1; j < n; j++) {
265 //             if (edges[i].weight < edges[j].weight) {
266 //                 swap(edges[i], edges[j]);
267 //             }
268 //         }
269 //     }
270 //     for (i = 0; i < n; i++) {
271 //         for (j = i + 1; j < n; j++) {
272 //             if (edges[i].weight < edges[j].weight) {
273 //                 swap(edges[i], edges[j]);
274 //             }
275 //         }
276 //     }
277 //     for (i = 0; i < n; i++) {
278 //         for (j = i + 1; j < n; j++) {
279 //             if (edges[i].weight < edges[j].weight) {
280 //                 swap(edges[i], edges[j]);
281 //             }
282 //         }
283 //     }
284 //     for (i = 0; i < n; i++) {
285 //         for (j = i + 1; j < n; j++) {
286 //             if (edges[i].weight < edges[j].weight) {
287 //                 swap(edges[i], edges[j]);
288 //             }
289 //         }
290 //     }
291 //     for (i = 0; i < n; i++) {
292 //         for (j = i + 1; j < n; j++) {
293 //             if (edges[i].weight < edges[j].weight) {
294 //                 swap(edges[i], edges[j]);
295 //             }
296 //         }
297 //     }
298 //     for (i = 0; i < n; i++) {
299 //         for (j = i + 1; j < n; j++) {
300 //             if (edges[i].weight < edges[j].weight) {
301 //                 swap(edges[i], edges[j]);
302 //             }
303 //         }
304 //     }
305 //     for (i = 0; i < n; i++) {
306 //         for (j = i + 1; j < n; j++) {
307 //             if (edges[i].weight < edges[j].weight) {
308 //                 swap(edges[i], edges[j]);
309 //             }
310 //         }
311 //     }
312 //     for (i = 0; i < n; i++) {
313 //         for (j = i + 1; j < n; j++) {
314 //             if (edges[i].weight < edges[j].weight) {
315 //                 swap(edges[i], edges[j]);
316 //             }
317 //         }
318 //     }
319 //     for (i = 0; i < n; i++) {
320 //         for (j = i + 1; j < n; j++) {
321 //             if (edges[i].weight < edges[j].weight) {
322 //                 swap(edges[i], edges[j]);
323 //             }
324 //         }
325 //     }
326 //     for (i = 0; i < n; i++) {
327 //         for (j = i + 1; j < n; j++) {
328 //             if (edges[i].weight < edges[j].weight) {
329 //                 swap(edges[i], edges[j]);
330 //             }
331 //         }
332 //     }
333 //     for (i = 0; i < n; i++) {
334 //         for (j = i + 1; j < n; j++) {
335 //             if (edges[i].weight < edges[j].weight) {
336 //                 swap(edges[i], edges[j]);
337 //             }
338 //         }
339 //     }
340 //     for (i = 0; i < n; i++) {
341 //         for (j = i + 1; j < n; j++) {
342 //             if (edges[i].weight < edges[j].weight) {
343 //                 swap(edges[i], edges[j]);
344 //             }
345 //         }
346 //     }
347 //     for (i = 0; i < n; i++) {
348 //         for (j = i + 1; j < n; j++) {
349 //             if (edges[i].weight < edges[j].weight) {
350 //                 swap(edges[i], edges[j]);
351 //             }
352 //         }
353 //     }
354 //     for (i = 0; i < n; i++) {
355 //         for (j = i + 1; j < n; j++) {
356 //             if (edges[i].weight < edges[j].weight) {
357 //                 swap(edges[i], edges[j]);
358 //             }
359 //         }
360 //     }
361 //     for (i = 0; i < n; i++) {
362 //         for (j = i + 1; j < n; j++) {
363 //             if (edges[i].weight < edges[j].weight) {
364 //                 swap(edges[i], edges[j]);
365 //             }
366 //         }
367 //     }
368 //     for (i = 0; i < n; i++) {
369 //         for (j = i + 1; j < n; j++) {
370 //             if (edges[i].weight < edges[j].weight) {
371 //                 swap(edges[i], edges[j]);
372 //             }
373 //         }
374 //     }
375 //     for (i = 0; i < n; i++) {
376 //         for (j = i + 1; j < n; j++) {
377 //             if (edges[i].weight < edges[j].weight) {
378 //                 swap(edges[i], edges[j]);
379 //             }
380 //         }
381 //     }
382 //     for (i = 0; i < n; i++) {
383 //         for (j = i + 1; j < n; j++) {
384 //             if (edges[i].weight < edges[j].weight) {
385 //                 swap(edges[i], edges[j]);
386 //             }
387 //         }
388 //     }
389 //     for (i = 0; i < n; i++) {
390 //         for (j = i + 1; j < n; j++) {
391 //             if (edges[i].weight < edges[j].weight) {
392 //                 swap(edges[i], edges[j]);
393 //             }
394 //         }
395 //     }
396 //     for (i = 0; i < n; i++) {
397 //         for (j = i + 1; j < n; j++) {
398 //             if (edges[i].weight < edges[j].weight) {
399 //                 swap(edges[i], edges[j]);
400 //             }
401 //         }
402 //     }
403 //     for (i = 0; i < n; i++) {
404 //         for (j = i + 1; j < n; j++) {
405 //             if (edges[i].weight < edges[j].weight) {
406 //                 swap(edges[i], edges[j]);
407 //             }
408 //         }
409 //     }
410 //     for (i = 0; i < n; i++) {
411 //         for (j = i + 1; j < n; j++) {
412 //             if (edges[i].weight < edges[j].weight) {
413 //                 swap(edges[i], edges[j]);
414 //             }
415 //         }
416 //     }
417 //     for (i = 0; i < n; i++) {
418 //         for (j = i + 1; j < n; j++) {
419 //             if (edges[i].weight < edges[j].weight) {
420 //                 swap(edges[i], edges[j]);
421 //             }
422 //         }
423 //     }
424 //     for (i = 0; i < n; i++) {
425 //         for (j = i + 1; j < n; j++) {
426 //             if (edges[i].weight < edges[j].weight) {
427 //                 swap(edges[i], edges[j]);
428 //             }
429 //         }
430 //     }
431 //     for (i = 0; i < n; i++) {
432 //         for (j = i + 1; j < n; j++) {
433 //             if (edges[i].weight < edges[j].weight) {
434 //                 swap(edges[i], edges[j]);
435 //             }
436 //         }
437 //     }
438 //     for (i = 0; i < n; i++) {
439 //         for (j = i + 1; j < n; j++) {
440 //             if (edges[i].weight < edges[j].weight) {
441 //                 swap(edges[i], edges[j]);
442 //             }
443 //         }
444 //     }
445 //     for (i = 0; i < n; i++) {
446 //         for (j = i + 1; j < n; j++) {
447 //             if (edges[i].weight < edges[j].weight) {
448 //                 swap(edges[i], edges[j]);
449 //             }
450 //         }
451 //     }
452 //     for (i = 0; i < n; i++) {
453 //         for (j = i + 1; j < n; j++) {
454 //             if (edges[i].weight < edges[j].weight) {
455 //                 swap(edges[i], edges[j]);
456 //             }
457 //         }
458 //     }
459 //     for (i = 0; i < n; i++) {
460 //         for (j = i + 1; j < n; j++) {
461 //             if (edges[i].weight < edges[j].weight) {
462 //                 swap(edges[i], edges[j]);
463 //             }
464 //         }
465 //     }
466 //     for (i = 0; i < n; i++) {
467 //         for (j = i + 1; j < n; j++) {
468 //             if (edges[i].weight < edges[j].weight) {
469 //                 swap(edges[i], edges[j]);
470 //             }
471 //         }
472 //     }
473 //     for (i = 0; i < n; i++) {
474 //         for (j = i + 1; j < n; j++) {
475 //             if (edges[i].weight < edges[j].weight) {
476 //                 swap(edges[i], edges[j]);
477 //             }
478 //         }
479 //     }
480 //     for (i = 0; i < n; i++) {
481 //         for (j = i + 1; j < n; j++) {
482 //             if (edges[i].weight < edges[j].weight) {
483 //                 swap(edges[i], edges[j]);
484 //             }
485 //         }
486 //     }
487 //     for (i = 0; i < n; i++) {
488 //         for (j = i + 1; j < n; j++) {
489 //             if (edges[i].weight < edges[j].weight) {
490 //                 swap(edges[i], edges[j]);
491 //             }
492 //         }
493 //     }
494 //     for (i = 0; i < n; i++) {
495 //         for (j = i + 1; j < n; j++) {
496 //             if (edges[i].weight < edges[j].weight) {
497 //                 swap(edges[i], edges[j]);
498 //             }
499 //         }
500 //     }
501 //     for (i = 0; i < n; i++) {
502 //         for (j = i + 1; j < n; j++) {
503 //             if (edges[i].weight < edges[j].weight) {
504 //                 swap(edges[i], edges[j]);
505 //             }
506 //         }
507 //     }
508 //     for (i = 0; i < n; i++) {
509 //         for (j = i + 1; j < n; j++) {
510 //             if (edges[i].weight < edges[j].weight) {
511 //                 swap(edges[i], edges[j]);
512 //             }
513 //         }
514 //     }
515 //     for (i = 0; i < n; i++) {
516 //         for (j = i + 1; j < n; j++) {
517 //             if (edges[i].weight < edges[j].weight) {
518 //                 swap(edges[i], edges[j]);
519 //             }
520 //         }
521 //     }
522 //     for (i = 0; i < n; i++) {
523 //         for (j = i + 1; j < n; j++) {
524 //             if (edges[i].weight < edges[j].weight) {
525 //                 swap(edges[i], edges[j]);
526 //             }
527 //         }
528 //     }
529 //     for (i = 0; i < n; i++) {
530 //         for (j = i + 1; j < n; j++) {
531 //             if (edges[i].weight < edges[j].weight) {
532 //                 swap(edges[i], edges[j]);
533 //             }
534 //         }
535 //     }
536 //     for (i = 0; i < n; i++) {
537 //         for (j = i + 1; j < n; j++) {
538 //             if (edges[i].weight < edges[j].weight) {
539 //                 swap(edges[i], edges[j]);
540 //             }
541 //         }
542 //     }
543 //     for (i = 0; i < n; i++) {
544 //         for (j = i + 1; j < n; j++) {
545 //             if (edges[i].weight < edges[j].weight) {
546 //                 swap(edges[i], edges[j]);
547 //             }
548 //         }
549 //     }
550 //     for (i = 0; i < n; i++) {
551 //         for (j = i + 1; j < n; j++) {
552 //             if (edges[i].weight < edges[j].weight) {
553 //                 swap(edges[i], edges[j]);
554 //             }
555 //         }
556 //     }
557 //     for (i = 0; i < n; i++) {
558 //         for (j = i + 1; j < n; j++) {
559 //             if (edges[i].weight < edges[j].weight) {
560 //                 swap(edges[i], edges[j]);
561 //             }
562 //         }
563 //     }
564 //     for (i = 0; i < n; i++) {
565 //         for (j = i + 1; j < n; j++) {
566 //             if (edges[i].weight < edges[j].weight) {
567 //                 swap(edges[i], edges[j]);
568 //             }
569 //         }
570 //     }
571 //     for (i = 0; i < n; i++) {
572 //         for (j = i + 1; j < n; j++) {
573 //             if (edges[i].weight < edges[j].weight) {
574 //                 swap(edges[i], edges[j]);
575 //             }
576 //         }
577 //     }
578 //     for (i = 0; i < n; i++) {
579 //         for (j = i + 1; j < n; j++) {
580 //             if (edges[i].weight < edges[j].weight) {
581 //                 swap(edges[i], edges[j]);
582 //             }
583 //         }
584 //     }
585 //     for (i = 0; i < n; i++) {
586 //         for (j = i + 1; j < n; j++) {
587 //             if (edges[i].weight < edges[j].weight) {
588 //                 swap(edges[i], edges[j]);
589 //             }
590 //         }
591 //     }
592 //     for (i = 0; i < n; i++) {
593 //         for (j = i + 1; j < n; j++) {
594 //             if (edges[i].weight < edges[j].weight) {
595 //                 swap(edges[i], edges[j]);
596 //             }
597 //         }
598 //     }
599 //     for (i = 0; i < n; i++) {
600 //         for (j = i + 1; j < n; j++) {
601 //             if (edges[i].weight < edges[j].weight) {
602 //                 swap(edges[i], edges[j]);
603 //             }
604 //         }
605 //     }
606 //     for (i = 0; i < n; i++) {
607 //         for (j = i + 1; j < n; j++) {
608 //             if (edges[i].weight < edges[j].weight) {
609 //                 swap(edges[i], edges[j]);
610 //             }
611 //         }
612 //     }
613 //     for (i = 0; i < n; i++) {
614 //         for (j = i + 1; j < n; j++) {
615 //             if (edges[i].weight < edges[j].weight) {
616 //                 swap(edges[i], edges[j]);
617 //             }
618 //         }
619 //     }
620 //     for (i = 0; i < n; i++) {
621 //         for (j = i + 1; j < n; j++) {
622 //             if (edges[i].weight < edges[j].weight) {
623 //                 swap(edges[i], edges[j]);
624 //             }
625 //         }
626 //     }
627 //     for (i = 0; i < n; i++) {
628 //         for (j = i + 1; j < n; j++) {
629 //             if (edges[i].weight < edges[j].weight) {
630 //                 swap(edges[i], edges[j]);
631 //             }
632 //         }
633 //     }
634 //     for (i = 0; i < n; i++) {
635 //         for (j = i + 1; j < n; j++) {
636 //             if (edges[i].weight < edges[j].weight) {
637 //                 swap(edges[i], edges[j]);
638 //             }
639 //         }
640 //     }
641 //     for (i = 0; i < n; i++) {
642 //         for (j = i + 1; j < n; j++) {
643 //             if (edges[i].weight < edges[j].weight) {
644 //                 swap(edges[i], edges[j]);
645 //             }
646 //         }
647 //     }
648 //     for (i = 0; i < n; i++) {
649 //         for (j = i + 1; j < n; j++) {
650 //             if (edges[i].weight < edges[j].weight) {
651 //                 swap(edges[i], edges[j]);
652 //             }
653 //         }
654 //     }
655 //     for (i = 0; i < n; i++) {
656 //         for (j = i + 1; j < n; j++) {
657 //             if (edges[i].weight < edges[j].weight) {
658 //                 swap(edges[i], edges[j]);
659 //             }
660 //         }
661 //     }
662 //     for (i = 0; i < n; i++) {
663 //         for (j = i + 1; j < n; j++) {
664 //             if (edges[i].weight < edges[j].weight) {
665 //                 swap(edges[i], edges[j]);
666 //             }
667 //         }
668 //     }
669 //     for (i = 0; i < n; i++) {
670 //         for (j = i + 1; j < n; j++) {
671 //             if (edges[i].weight < edges[j].weight) {
672 //                 swap(edges[i], edges[j]);
673 //             }
674 //         }
675 //     }
676 //     for (i = 0; i < n; i++) {
677 //         for (j = i + 1; j < n; j++) {
678 //             if (edges[i].weight < edges[j].weight) {
679 //                 swap(edges[i], edges[j]);
680 //             }
681 //         }
682 //     }
683 //     for (i = 0; i < n; i++) {
684 //         for (j = i + 1; j < n; j++) {
685 //             if (edges[i].weight < edges[j].weight) {
686 //                 swap(edges[i], edges[j]);
687 //             }
688 //         }
689 //     }
690 //     for (i = 0; i < n; i++) {
691 //         for (j = i + 1; j < n; j++) {
692 //             if (edges[i].weight < edges[j].weight) {
693 //                 swap(edges[i], edges[j]);
694 //             }
695 //         }
696 //     }
697 //     for (i = 0; i < n; i++) {
698 //         for (j = i + 1; j < n; j++) {
699 //             if (edges[i].weight < edges[j].weight) {
700 //                 swap(edges[i], edges[j]);
701 //             }
702 //         }
703 //     }
704 //     for (i = 0; i < n; i++) {
705 //         for (j = i + 1; j < n; j++) {
706 //             if (edges[i].weight < edges[j].weight) {
707 //                 swap(edges[i], edges[j]);
708 //             }
709 //         }
710 //     }
711 //     for (i = 0; i < n; i++) {
712 //         for (j = i + 1; j < n; j++) {
713 //             if (edges[i].weight < edges[j].weight) {
714 //                 swap(edges[i], edges[j]);
715 //             }
716 //         }
717 //     }
718 //     for (i = 0; i < n; i++) {
719 //         for (j = i + 1; j < n; j++) {
720 //             if (edges[i].weight < edges[j].weight) {
721 //                 swap(edges[i], edges[j]);
722 //             }
723 //         }
724 //     }
725 //     for (i = 0; i < n; i++) {
726 //         for (j = i + 1; j < n; j++) {
727 //             if (edges[i].weight < edges[j].weight) {
728 //                 swap(edges[i], edges[j]);
729 //             }
730 //         }
731 //     }
732 //     for (i = 0; i < n; i++) {
733 //         for (j = i + 1; j < n; j++) {
734 //             if (edges[i].weight < edges[j].weight) {
735 //                 swap(edges[i], edges[j]);
736 //             }
737 //         }
738 //     }
739 //     for (i = 0; i < n; i++) {
740 //         for (j = i + 1; j < n; j++) {
741 //             if (edges[i].weight < edges[j].weight) {
742 //                 swap(edges[i], edges[j]);
743 //             }
744 //         }
745 //     }
746 //     for (i = 0; i < n; i++) {
747 //         for (j = i + 1; j < n; j++) {
748 //             if (edges[i].weight < edges[j].weight) {
749 //                 swap(edges[i], edges[j]);
750 //             }
751 //         }
752 //     }
753 //     for (i = 0; i < n; i++) {
754 //         for (j = i + 1; j < n; j++) {
755 //             if (edges[i].weight < edges[j].weight) {
756 //                 swap(edges[i], edges[j]);
757 //             }
758 //         }
759 //     }
760 //     for (i = 0; i < n; i++) {
761 //         for (j = i + 1; j < n; j++) {
762 //             if (edges[i].weight < edges[j].weight) {
763 //                 swap(edges[i], edges[j]);
764 //             }
765 //         }
766 //     }
767 //     for (i = 0; i < n; i++) {
768 //         for (j = i + 1; j < n; j++) {
769 //             if (edges[i].weight < edges[j].weight) {
770 //                 swap(edges[i], edges[j]);
771 //             }
772 //         }
773 //     }
774 //     for (i = 0; i < n; i++) {
775 //         for (j = i + 1; j < n; j++) {
776 //             if (edges[i].weight < edges[j].weight) {
777 //                 swap(edges[i], edges[j]);
778 //             }
779 //         }
780 //     }
781 //     for (i = 0; i < n; i++) {
782 //         for (j = i + 1; j < n; j++) {
783 //             if (edges[i].weight < edges[j].weight) {
784 //                 swap(edges[i], edges[j]);
785 //             }
786 //         }
787 //     }
788 //     for (i = 0; i < n; i++) {
789 //         for (j = i + 1; j < n; j++) {
790 //             if (edges[i].weight < edges[j].weight) {
791 //                 swap(edges[i], edges[j]);
792 //             }
793 //         }
794 //     }
795 //     for (i = 0; i < n; i++) {
796 //         for (j = i + 1;
```

21.

```

1 #include <stdio.h>
2 #include <limits.h>
3
4 int sum(int freq[], int i, int j) {
5     int s = 0;
6     for (int k = i; k <= j; k++)
7         s += freq[k];
8     return s;
9 }
10
11 int optimalBST(int keys[], int freq[], int n) {
12     int cost[n][n];
13
14     for (int i = 0; i < n; i++)
15         cost[i][i] = freq[i];
16
17     for (int L = 2; L <= n; L++) {
18         for (int i = 0; i <= n - L; i++) {
19             int j = i + L - 1;
20             cost[i][j] = INT_MAX;
21
22             for (int r = i; r <= j; r++) {
23                 int c = ((r > i) ? cost[i][r - 1] : 0) +
24                     ((r < j) ? cost[r + 1][j] : 0) +
25                     sum(freq, i, j);
26                 if (c < cost[i][j])
27                     cost[i][j] = c;
28             }
29         }
30     }
31
32     return cost[0][n - 1];
33 }
34
35 int main() {
36     int keys[] = {10, 12, 20};
37     int freq[] = {34, 0, 50};
38     int n = sizeof(keys) / sizeof(keys[0]);
39     printf("Cost of optimal BST is: %d", optimalBST(keys, freq, n));
40     return 0;
41 }

```

Cost of optimal BST is: 142

Process exited after 1.086 seconds with return value 0

Press any key to continue . . .

22.

```

1 #include <stdio.h>
2
3 int binomialCoeff(int n, int k) {
4     int c[n][k + 1];
5     int i, j;
6
7     for (i = 0; i <= n; i++) {
8         for (j = 0; j <= k; j++) {
9             if (i == 0 || j == 0)
10                 c[i][j] = 1;
11             else
12                 c[i][j] = c[i - 1][j - 1] + c[i - 1][j];
13         }
14     }
15
16     return c[n][k];
17 }
18
19 int main() {
20     int n = 5, k = 2;
21     printf("Binomial Coefficient C(%d, %d) is: %d", n, k, binomialCoeff(n, k));
22     return 0;
23 }

```

Binomial Coefficient C(5, 2) is: 10

Process exited after 1.2 seconds with return value 0

Press any key to continue . . .

23.

```

1 #include <stdio.h>
2
3 int main() {
4
5     int n, reverse = 0, remainder;
6
7     printf("Enter an integer: ");
8     scanf("%d", &n);
9
10    while (n != 0) {
11        remainder = n % 10;
12        reverse = reverse * 10 + remainder;
13        n /= 10;
14    }
15
16    printf("Reversed number = %d", reverse);
17
18    return 0;
19 }

```

Enter an integer: 12345
Reversed number = 54321

Process exited after 4.859 seconds with return value 0
Press any key to continue . . .

Compiler: 0 Errors, 0 Warnings
Output Filename: C:\Users\nagaj\Documents\DA\EXP-24.exe
Output Size: 128.1015625 KiB
Compilation Time: 0.10s

24.

```

1 #include <stdio.h>
2
3 int isPerfectNumber(int num) {
4     int sum = 0;
5     for (int i = 1; i <= num / 2; i++) {
6         if (num % i == 0) {
7             sum += i;
8         }
9     }
10    return sum == num;
11 }
12
13 int main() {
14     int num = 28;
15     if (isPerfectNumber(num)) {
16         printf("%d is a perfect number.", num);
17     } else {
18         printf("%d is not a perfect number.", num);
19     }
20     return 0;
21 }

```

28 is a perfect number.

Process exited after 1.912 seconds with return value 0
Press any key to continue . . .

Compiler: 0 Errors, 0 Warnings
Output Filename: C:\Users\nagaj\Documents\DA\EXP-25.exe
Output Size: 128.4697265625 KiB
Compilation Time: 0.20s

25.

```

1 #include <stdio.h>
2 #include <limits.h>
3
4 #define V 4
5
6 int tsp(int graph[V][V], int mask, int pos, int dp[][1 << V]) {
7     if (mask == (1 << V) - 1)
8         return graph[pos][0];
9     if (dp[pos][mask] != -1)
10         return dp[pos][mask];
11
12     int minCost = INT_MAX;
13
14     for (int city = 0; city < V; city++) {
15         if ((mask & (1 << city)) == 0) {
16             int newCost = graph[pos][city] + tsp(graph, mask | (1 << city), city, dp);
17             minCost = (newCost < minCost) ? newCost : minCost;
18         }
19     }
20
21     return dp[pos][mask] = minCost;
22 }
23
24 int main() {
25     int graph[V][V] = {
26         {0, 10, 15, 20},
27         {10, 0, 35, 25},
28         {15, 35, 0, 30},
29         {20, 25, 30, 0}
30     };
31
32     int dp[V][1 << V];
33     for (int i = 0; i < V; i++) {
34         for (int j = 0; j < (1 << V); j++) {
35             dp[i][j] = -1;
36         }
37     }
38
39     int minCost = tsp(graph, 1, 0, dp);
40     printf("Minimum cost for the Travelling Salesman Problem is: %d\n", minCost);
41     return 0;
42 }

```

Minimum cost for the Travelling Salesman Problem is: 80

Process exited after 1.971 seconds with return value 0
Press any key to continue . . .

26.

```

1 #include <stdio.h>
2
3 void printPattern(int n) {
4     if (n == 0) {
5         return;
6     }
7     printPattern(n - 1);
8     for (int i = 1; i <= n; i++) {
9         printf("%d", i);
10    }
11    printf("\n");
12 }
13
14 int main() {
15     int rows = 4;
16     printPattern(rows);
17     return 0;
18 }

```

1
12
123
1234

Process exited after 3.07 seconds with return value 0
Press any key to continue . . .

27.

The screenshot shows a C++ IDE with a project named 'EXP-29.cpp'. The code implements the Floyd-Warshall algorithm to find the shortest distances between every pair of vertices in a graph. The output window displays the following shortest distances:

```

Shortest distances between every pair of vertices:
0      5      8      9
INF    0      3      4
INF    INF    0      1
INF    INF    INF    0
  
```

Below the output, it states: "Process exited after 0.7291 seconds with return value 0. Press any key to continue . . .".

28.

The screenshot shows a C++ IDE with a project named 'EXP-30.cpp'. The code implements a program that prints a Pascal's triangle. The output window displays the following Pascal's triangle for 4 rows:

```

Enter the number of rows: 4
      1
     1 1
    1 2 1
   1 3 3 1
  
```

Below the output, it states: "Process exited after 6.434 seconds with return value 0. Press any key to continue . . .".

29.

The screenshot shows a C++ IDE with a project named 'EXP-29.cpp'. The code implements a linked list with an 'insert' function and a 'print_list' function. The main function inserts the values 5, 10, and 15 into the list and then prints it. The output window shows the list after insertion: '15 -> 10 -> 5 -> NULL'. The process exited after 1.385 seconds with a return value of 0.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* next;
7 };
8
9 void insert_number(struct Node** head, int value) {
10     struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
11     new_node->data = value;
12     new_node->next = *head;
13     *head = new_node;
14 }
15
16 void print_list(struct Node* head) {
17     struct Node* temp = head;
18     while (temp != NULL) {
19         printf("%d -> ", temp->data);
20         temp = temp->next;
21     }
22     printf("NULL\n");
23 }
24
25 int main() {
26     struct Node* head = NULL;
27     insert_number(&head, 5);
28     insert_number(&head, 10);
29     insert_number(&head, 15);
30
31     printf("List after insertion: ");
32     print_list(head);
33     return 0;
34 }

```

Output:

```

List after insertion: 15 -> 10 -> 5 -> NULL
Process exited after 1.385 seconds with return value 0
Press any key to continue . . .

```

30.

The screenshot shows a C++ IDE with a project named 'EXP-30.cpp'. The code implements a recursive function 'sumOfDigits' to calculate the sum of the digits of a number. The main function calls this function with the number 12345 and prints the result. The output window shows the sum of digits of 12345 is 15. The process exited after 1.637 seconds with a return value of 0.

```

1 #include <stdio.h>
2
3 int sumOfDigits(int num) {
4     if (num == 0)
5         return 0;
6     return (num % 10) + sumOfDigits(num / 10);
7 }
8
9 int main() {
10     int number = 12345;
11     int sum = sumOfDigits(number);
12     printf("Sum of digits of %d is: %d\n", number, sum);
13     return 0;
14 }

```

Output:

```

Sum of digits of 12345 is: 15
Process exited after 1.637 seconds with return value 0
Press any key to continue . . .

```

31.

```

1 #include <iostream.h>
2 #include <stdio.h>
3
4 void findMinimumMaximum(int arr[], int n)
5 {
6     int i;
7     int minE = INT_MAX, maxE = INT_MIN;
8
9     for (i = 0; i < n; i++) {
10         if (arr[i] < minE) {
11             minE = arr[i];
12         }
13         if (arr[i] > maxE) {
14             maxE = arr[i];
15         }
16     }
17
18     printf("The minimum element is %d", minE);
19     printf("\n");
20     printf("The maximum element is %d", maxE);
21     return;
22 }
23
24 int main()
25 {
26     int arr[] = { 1, 2, 4, -1 };
27     int n = sizeof(arr) / sizeof(arr[0]);
28     findMinimumMaximum(arr, n);
29     return 0;
30 }

```

Output:

```

The minimum element is -1
The maximum element is 4
-----
Process exited after 0.8437 seconds with return value 0
Press any key to continue . . .

```

32.

```

1 #include <iostream.h>
2 #include <stdio.h>
3
4 #define N 4
5
6 void printSolution(int board[N][N]) {
7     for (int i = 0; i < N; i++) {
8         for (int j = 0; j < N; j++) {
9             printf("%d ", board[i][j]);
10         }
11         printf("\n");
12     }
13 }
14
15 bool isSafe(int board[N][N], int row, int col) {
16     int i;
17     for (i = 0; i < row; i++) {
18         if (board[i][col])
19             return false;
20     }
21     for (i = row, j = col; i >= 0 && j >= 0; i--, j--) {
22         if (board[i][j])
23             return false;
24     }
25     for (i = row, j = col; i <= N-1 && j <= N-1; i++, j--) {
26         if (board[i][j])
27             return false;
28     }
29     return true;
30 }
31
32 bool solveQueensUtil(int board[N][N], int col) {
33     if (col >= N)
34         return true;
35     for (int i = 0; i < N; i++) {
36         if (isSafe(board, i, col)) {
37             board[i][col] = 1;
38             if (solveQueensUtil(board, col + 1))
39                 return true;
40             board[i][col] = 0;
41         }
42     }
43     return false;
44 }
45
46 bool solveQueens() {
47     int board[N][N] = {
48         {0, 0, 0, 0},
49         {0, 0, 0, 0},
50         {0, 0, 0, 0},
51         {0, 0, 0, 0}
52     };
53     if (solveQueensUtil(board, 0) == false) {
54         printf("Solution does not exist");
55         return false;
56     }
57     printSolution(board);
58     return true;
59 }
60
61 int main() {
62     solveQueens();
63     return 0;
64 }

```

Output:

```

0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
-----
Process exited after 0.7386 seconds with return value 0
Press any key to continue . . .

```


33.

File Edit Search View Project Execute Tools ASStyle Window Help

TDH-GCC 4.9.2 (64-bit Release)

Project Classes Debug

EXP-33.cpp

```

1 #include <stdio.h>
2
3 int isSubsetSum(int set[], int n, int sum) {
4     if (sum == 0) return 1;
5     if (n == 0) return 0;
6     if (set[n-1] > sum) return isSubsetSum(set, n-1, sum);
7     return isSubsetSum(set, n-1, sum) || isSubsetSum(set, n-1, sum - set[n-1]);
8 }
9
10 int main() {
11     int set[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
12     int n = sizeof(set) / sizeof(set[0]);
13     int sum = 15;
14     if (isSubsetSum(set, n, sum)) printf("Found a subset with given sum");
15     else printf("No subset with given sum");
16     return 0;
17 }

```

Output: 1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10

Process exited after 1.016 seconds with return value 0
Press any key to continue . . .

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation

Output Size: 128.103515625 KkB
Compilation Time: 0.19s

Shorten compiler paths

Line: 24 Col: 1 Sel: 0 Lines: 31 Length: 406 Insert Done parsing in 0.015 seconds

34.

File Edit Search View Project Execute Tools ASStyle Window Help

TDH-GCC 4.9.2 (64-bit Release)

Project Classes Debug

EXP-34.cpp

```

1 #include <stdio.h>
2 int isSubsetSum(int set[], int n, int sum) {
3     if (sum == 0) return 1;
4     if (n == 0) return 0;
5     if (set[n-1] > sum) return isSubsetSum(set, n-1, sum);
6     return isSubsetSum(set, n-1, sum) || isSubsetSum(set, n-1, sum - set[n-1]);
7 }
8
9 int main() {
10     int set[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
11     int n = sizeof(set) / sizeof(set[0]);
12     int sum = 15;
13     if (isSubsetSum(set, n, sum)) printf("Found a subset with given sum");
14     else printf("No subset with given sum");
15     return 0;
16 }

```

Output: Found a subset with given sum

Process exited after 0.6903 seconds with return value 0
Press any key to continue . . .

Line: 15 Col: 2 Sel: 0 Lines: 15 Length: 522 Insert Done parsing in 0.047 seconds

35.

The screenshot shows a C++ IDE with a project named 'EXP-35.cpp'. The code implements a graph coloring algorithm using a recursive function 'bool isColorable(int v, int color[], int n, int color[])'. The main function 'main()' reads the number of vertices 'n' and the edges of the graph, then calls 'isColorable()' to find a valid coloring. The output window shows the following text:

```

Solution Exists: Following are the assigned colors
1 2 3 2

Process exited after 1.675 seconds with return value 0
Press any key to continue . . .

```

36.

The screenshot shows a C++ IDE with a project named 'EXP-36.cpp'. The code implements a container loading algorithm using a recursive function 'bool container_loader(int k)'. The main function 'main()' reads the number of containers 'n', the weights of the containers, and the capacity of the loader, then calls 'container_loader()' to find a valid solution. The output window shows the following text:

```

Enter the number of containers: 4
Enter the weights of the containers:
2
2
4
5
Enter the capacity of the loader: 2
Solution: 2
Solution: 2
Solution: 2

Process exited after 60.34 seconds with return value 0
Press any key to continue . . .

```

37.

```

1 #include <stdio.h>
2
3 void generate_factors(int n, int i) {
4     if (i > n)
5         return;
6     if (n % i == 0) {
7         printf("%d ", i);
8     }
9     generate_factors(n, i + 1);
10 }
11
12 int main() {
13     int n;
14     printf("Enter the value of n: ");
15     scanf("%d", &n);
16     printf("Factors of %d are: ", n);
17     generate_factors(n, 1);
18     return 0;
19 }

```

Output Window:

```

Enter the value of n: 5
Factors of 5 are: 1 5
-----
Process exited after 6.583 seconds with return value 0
Press any key to continue . . .

```

Compiler Log:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagal\Documents\DA\EXP-38.exe
- Output Size: 128.6416015625 KiB
- Compilation Time: 0.17s

```

Line: 23 Col: 2 Sel: 0 Lines: 23 Length: 374 Insert Done parsing in 0.016 seconds

38.

```

1 #include <stdio.h>
2 #include <limits.h>
3
4 #define n 4
5
6 int cost[n][n] = {
7     {10, 2, 4, 5},
8     {5, 15, 7, 12},
9     {6, 8, 4, 1},
10    {4, 7, 2, 16}
11 };
12
13 int min_cost = INT_MAX;
14 int assigned[n];
15 int visited[n] = {0};
16
17 void assign_task(int worker, int total_cost) {
18     if (worker == n) {
19         if (total_cost < min_cost) {
20             min_cost = total_cost;
21             for (int i = 0; i < n; i++) {
22                 assigned[i] = visited[i];
23             }
24         }
25         return;
26     }
27     for (int task = 0; task < n; task++) {
28         if (!visited[task]) {
29             visited[task] = 1;
30             assign_task(worker + 1, total_cost + cost[worker][task]);
31             visited[task] = 0;
32         }
33     }
34 }
35
36 int main() {
37     assign_task(0, 0);
38     printf("Minimum Cost: %d\n", min_cost);
39     printf("Assignment: ");
40     for (int i = 0; i < n; i++) {
41         printf("Worker %d -> Task %d, ", i + 1, assigned[i] + 1);
42     }
43     printf("\n");
44     return 0;
45 }

```

Output Window:

```

Minimum Cost: 8
Assignment: Worker 1 -> Task 2, Worker 2 -> Task 2, Worker 3 -> Task 2, Worker 4 -> Task 2,
-----
Process exited after 0.5249 seconds with return value 0
Press any key to continue . . .

```

Compiler Log:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagal\Documents\DA\EXP-38.exe
- Output Size: 128.6416015625 KiB
- Compilation Time: 0.17s

```

Line: 45 Col: 18 Sel: 0 Lines: 48 Length: 1010 Insert Done parsing in 0 seconds

39.

The screenshot shows the Dev-C++ IDE with a C++ program for linear search. The code is as follows:

```

1 #include <stdio.h>
2
3 int linearSearch(int arr[], int n, int key) {
4     for (int i = 0; i < n; i++) {
5         if (arr[i] == key) {
6             return i;
7         }
8     }
9     return -1;
10 }
11
12 int main() {
13     int arr[] = {10, 20, 30, 40, 50};
14     int key = 30;
15     int n = sizeof(arr) / sizeof(arr[0]);
16     int result = linearSearch(arr, n, key);
17     if (result != -1) {
18         printf("Element found at index: %d\n", result);
19     } else {
20         printf("Element not found\n");
21     }
22     return 0;
23 }

```

The execution output window shows the following text:

```

C:\Users\naga\Documents\ID...
Element found at index: 2
-----
Process exited after 0.7782 seconds with return value 0
Press any key to continue . . .

```

The compiler output window shows the following text:

```

-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\naga\Documents\ID\EXP-32.exe
- Output Size: 128,630,671,975 KiB
- Compilation Time: 0.20s

```

40.

The screenshot shows the Dev-C++ IDE with a C++ program for linked list insertion. The code is as follows:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* next;
7 };
8
9 void insert_number(struct Node** head, int value) {
10     struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
11     new_node->data = value;
12     new_node->next = *head;
13     *head = new_node;
14 }
15
16 void print_list(struct Node* head) {
17     struct Node* temp = head;
18     while (temp != NULL) {
19         printf("%d -> ", temp->data);
20         temp = temp->next;
21     }
22     printf("NULL\n");
23 }
24
25 int main() {
26     struct Node* head = NULL;
27
28     insert_number(&head, 5);
29     insert_number(&head, 10);
30     insert_number(&head, 15);
31
32     printf("List after insertion: ");
33     print_list(head);
34
35     return 0;
36 }

```

The execution output window shows the following text:

```

C:\Users\naga\Documents\ID...
List after insertion: 15 -> 10 -> 5 -> NULL
-----
Process exited after 1.872 seconds with return value 0
Press any key to continue . . .

```

The compiler output window shows the following text:

```

-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\naga\Documents\ID\EXP-40.exe
- Output Size: 128,601,404,625 KiB
- Compilation Time: 0.17s

```