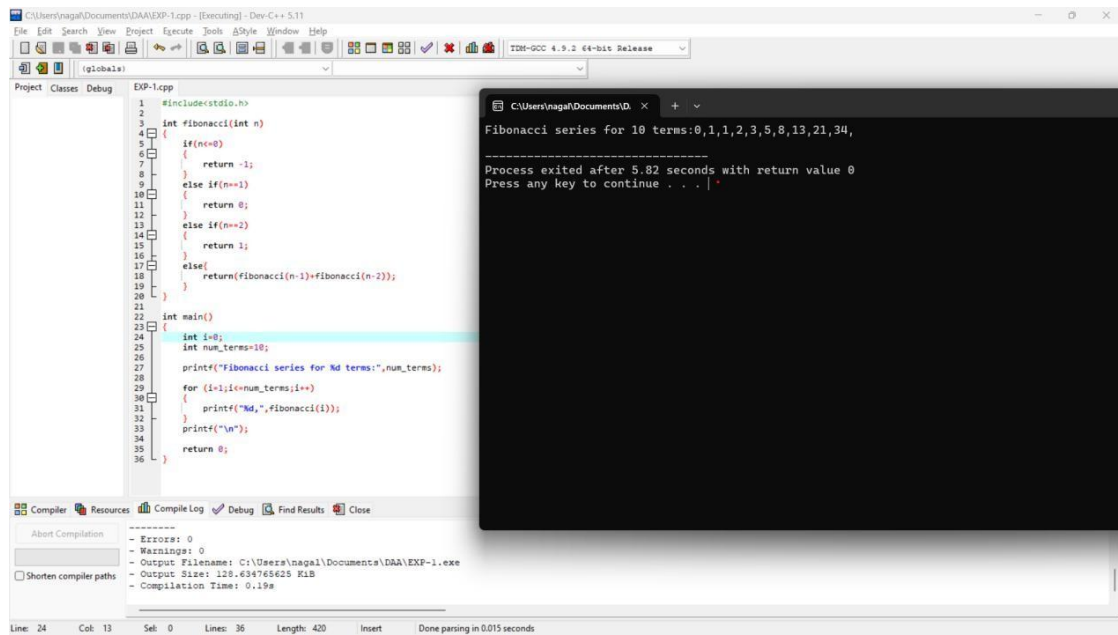


CSA0653-DESIGN AND ANALYSIS OF ALGORITHMS

1.



The screenshot shows a C++ IDE with a file named `EXP-1.cpp`. The code implements a recursive function `fibonacci(int n)` to calculate the nth term of the Fibonacci series. The `main` function calls `fibonacci(10)` and prints the result. The output window displays the Fibonacci series for 10 terms: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34. The process exited after 5.82 seconds with a return value of 0.

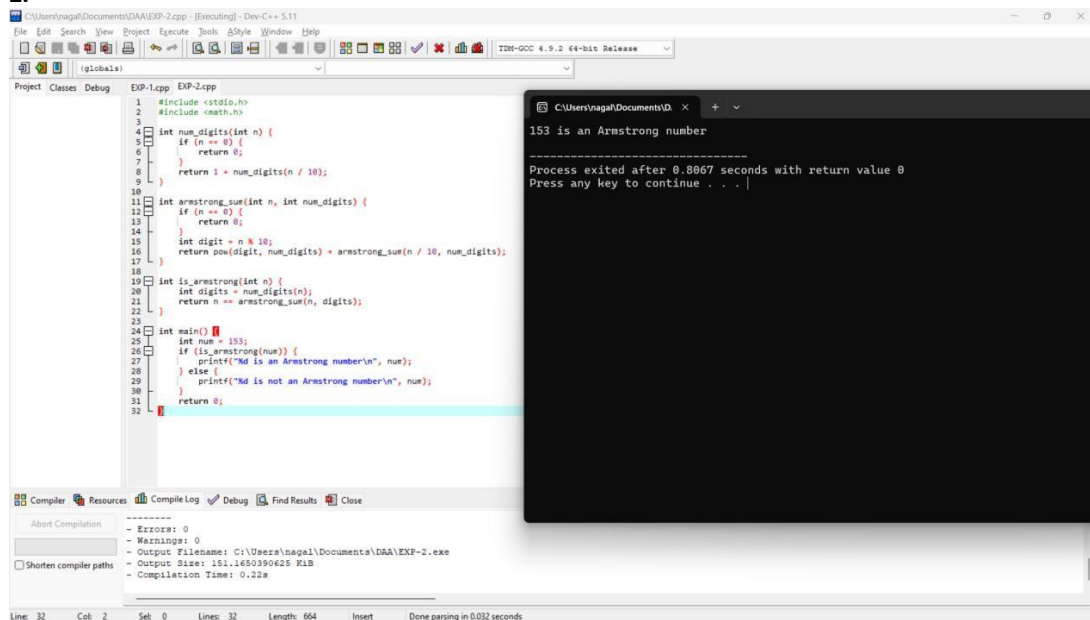
```
#include <stdio.h>

int fibonacci(int n)
{
    if(n==0)
        return 0;
    else if(n==1)
        return 1;
    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 a function `is_armstrong(int n)` to check if a number is an Armstrong number. The `main` function calls `is_armstrong(153)` and prints the result. The output window displays "153 is an Armstrong number". The process exited after 0.8867 seconds with a return value of 0.

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

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

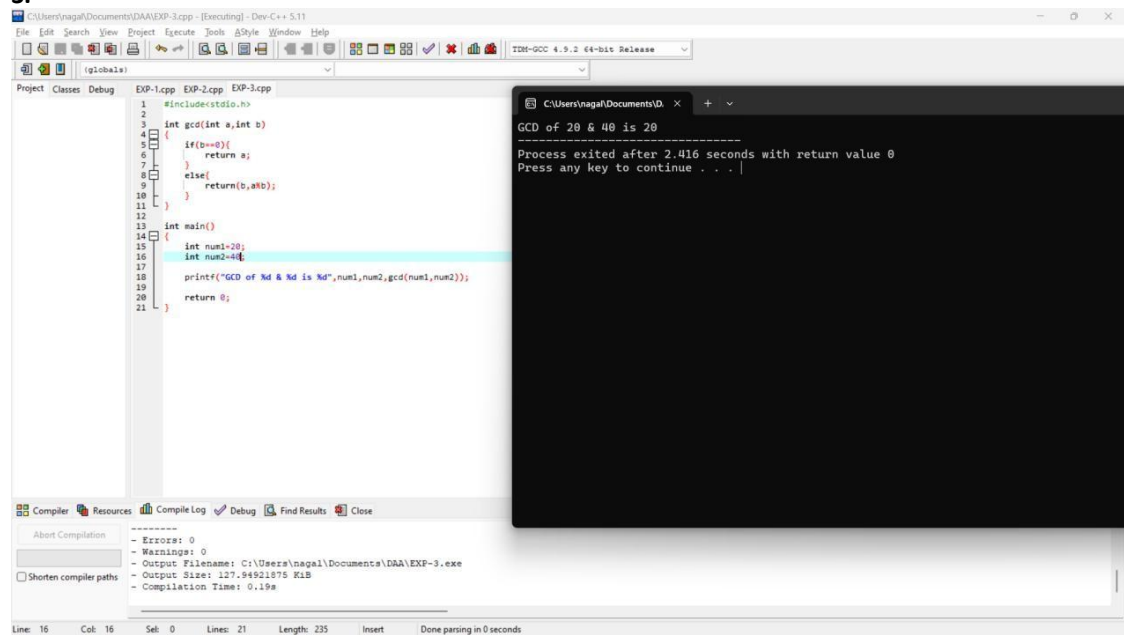
int armstrong_sum(int n, int num_digits)
{
    if(n==0)
        return 0;
    int digit = n % 10;
    return pow(digit, num_digits) + armstrong_sum(n / 10, num_digits);
}

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

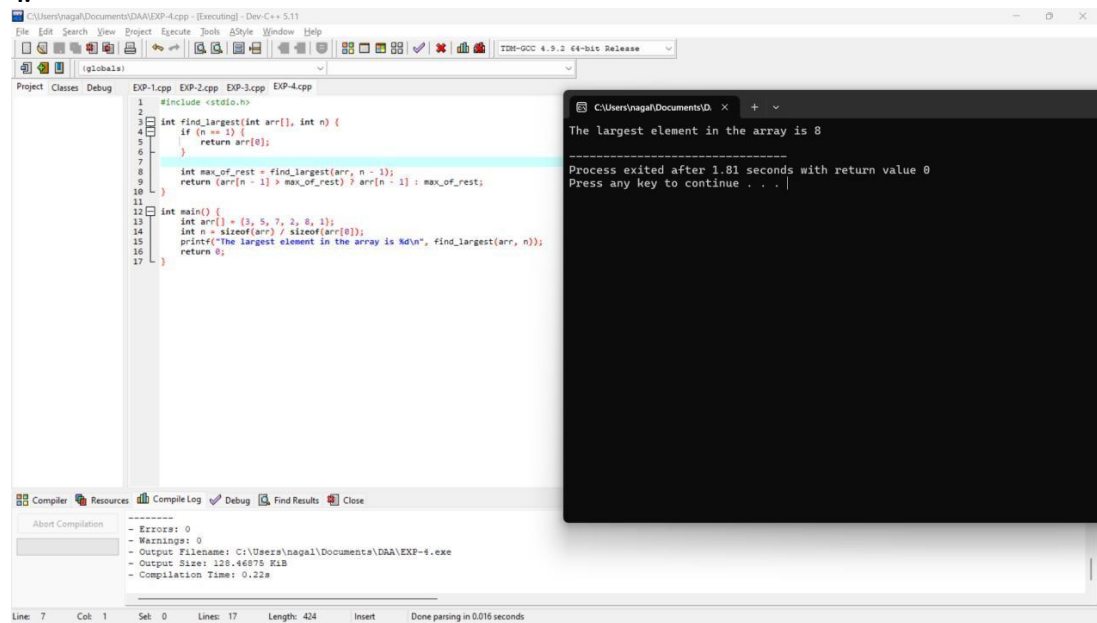
int main()
{
    int num = 153;
    if(is_armstrong(num))
        printf("153 is an Armstrong number\n");
    else
        printf("153 is not an Armstrong number\n");
    return 0;
}
```

Output: 153 is an Armstrong number Process exited after 0.8867 seconds with return value 0 Press any key to continue . . .

3.



4.



5.

The screenshot shows a C++ IDE with a project named 'EXP-5.cpp'. The code defines a recursive function 'factorial' and a 'main' function that calculates the factorial of 5. The output window shows the result: 'Factorial of 5 is 120'.

```

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     printf("Factorial of %d is %d",num,factorial(num));
15     return 0;
16 }
17
18

```

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

Compiler Output: Errors: 0, Warnings: 0, Output Filename: C:\Users\nagal\Documents\DA\EXP-5.exe, Output Size: 128.46289625 KIB, Compilation Time: 0.19s

6.

The screenshot shows a C++ IDE with a project named 'EXP-6.cpp'. The code defines a 'main' function that prompts the user for a number and checks if it is prime. The output window shows the user input '2' and the result: '2 is a prime number...'.

```

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

```

Output: Enter the number:2
2 is a prime number...
Process exited after 7.386 seconds with return value 0
Press any key to continue . . .

Compiler Output: Errors: 0, Warnings: 0, Output Filename: C:\Users\nagal\Documents\DA\EXP-6.exe, Output Size: 128.6015625 KIB, Compilation Time: 0.20s

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: TDM-GCC 4.9.2 64-bit Release
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagal\Documents\DA\EXP-7.exe
Output Size: 128,642,578,125 KiB
Compilation Time: 0.25s

Line: 11 Col: 9 Sel: 0 Lines: 28 Length: 642 Insert Done parsing in 0 seconds

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
16 int main() {
17     int arr[] = {64, 34, 25, 12, 22, 11, 90};
18     int n = sizeof(arr)/sizeof(arr[0]);
19     bubble_sort(arr, n);
20     printf("Sorted array: ");
21     for (int i = 0; i < n; i++) {
22         printf("%d ", arr[i]);
23     }
24     printf("\n");
25     return 0;
26 }

```

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: TDM-GCC 4.9.2 64-bit Release
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagal\Documents\DA\EXP-8.exe
Output Size: 128,639,648,375 KiB
Compilation Time: 0.17s

Line: 14 Col: 1 Sel: 0 Lines: 25 Length: 592 Insert Done parsing in 0 seconds

9.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 void readMatrix(int **mat, int rows, int cols) {
6     for (int i = 0; i < rows; i++) {
7         for (int j = 0; j < cols; j++) {
8             scanf("%d", &mat[i][j]);
9         }
10    }
11}
12
13 void printMatrix(int **mat, int rows, int cols) {
14     for (int i = 0; i < rows; i++) {
15         for (int j = 0; j < cols; j++) {
16             printf("%d ", mat[i][j]);
17         }
18         printf("\n");
19     }
20}
21
22 void multiplyMatrix(int **mat1, int rows1, int cols1, int **mat2, int rows2, int cols2, int **result) {
23     for (int i = 0; i < rows1; i++) {
24         for (int j = 0; j < cols2; j++) {
25             result[i][j] = 0;
26             for (int k = 0; k < cols1; k++) {
27                 result[i][j] += mat1[i][k] * mat2[k][j];
28             }
29         }
30     }
31}
32
33 int main() {
34     int rows1, cols1, rows2, cols2;
35     printf("Enter rows and columns for the first matrix: ");
36     scanf("%d %d", &rows1, &cols1);
37     printf("Enter rows and columns for the second matrix: ");
38     scanf("%d %d", &rows2, &cols2);
39     if (cols1 != rows2) {
40         printf("Error: Number of columns in the first matrix must be equal to the number of rows in the second matrix.\n");
41         return 1;
42     }
43     printf("Enter the first matrix:\n");
44     int **mat1 = (int **) malloc(rows1 * sizeof(int *));
45     for (int i = 0; i < rows1; i++) mat1[i] = (int *) malloc(cols1 * sizeof(int));
46     readMatrix(mat1, rows1, cols1);
47     printf("Enter the second matrix:\n");
48     int **mat2 = (int **) malloc(rows2 * sizeof(int *));
49     for (int i = 0; i < rows2; i++) mat2[i] = (int *) malloc(cols2 * sizeof(int));
50     readMatrix(mat2, rows2, cols2);
51     multiplyMatrix(mat1, rows1, cols1, mat2, rows2, cols2, result);
52     printf("Resultant matrix:\n");
53     printMatrix(result, rows1, cols2);
54     return 0;
55}

```

Compiler: GCC 4.9.2 64-bit Release

Errors: 0
Warnings: 0
Output Filename: C:\Users\nagaj\Documents\DA\EXP-9.exe
Output Size: 130.5771494375 KIB
Compilation Time: 0.19s

Line: 45 Col: 18 Set: 0 Lines: 61 Length: 1702 Insert Done parsing in 0 seconds

```

C:\Users\nagaj\Documents\DA\EXP-9.exe
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 <stdio.h>
2 #include <stdlib.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
15 int main() {
16     char string[] = "racecar";
17     int length = strlen(string);
18     if (is_palindrome(string, 0, length - 1)) {
19         printf("%s is a palindrome\n", string);
20     } else {
21         printf("%s is not a palindrome\n", string);
22     }
23     return 0;
24}

```

Compiler: GCC 4.9.2 64-bit Release

Errors: 0
Warnings: 0
Output Filename: C:\Users\nagaj\Documents\DA\EXP-10.exe
Output Size: 128.470703125 KIB
Compilation Time: 0.25s

Line: 8 Col: 6 Set: 0 Lines: 25 Length: 574 Insert Done parsing in 0.083 seconds

```

C:\Users\nagaj\Documents\DA\EXP-10.exe
racecar is a palindrome

Process exited after 2.886 seconds with return value 0
Press any key to continue . . .

```

11.

The screenshot shows a C++ IDE with a project named 'EXP-10.cpp'. The code defines a `copyString` function and a `main` function. The `main` function prompts the user to enter a string, reads it, and then prints both the source and copied strings. The output window shows the user input 'sai' and the program's output 'Source string: sai' and 'Copied string: sai'. The compiler log indicates a successful compilation with no errors or warnings.

```

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     printf("Enter a string to copy: ");
15     fgets(source, sizeof(source), stdin);
16     copyString(source, destination, 0);
17     printf("Source string: %s\n", source);
18     printf("Copied string: %s\n", destination);
19     return 0;
20 }

```

Compiler Output:

```

-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-11.exe
- Output Size: 128.638671875 KIB
- Compilation Time: 0.19s

```

12.

The screenshot shows a C++ IDE with a project named 'EXP-10.cpp'. The code implements a binary search algorithm. The `main` function initializes an array of 20 elements, sets a key to 12, and calls the `binarySearch` function. The output window shows 'Element found at index 5.' and the program's exit message. The compiler log indicates a successful compilation with no errors or warnings.

```

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

```

Compiler Output:

```

-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-12.exe
- Output Size: 128.6396484375 KIB
- Compilation Time: 0.20s

```

13.

Dev-C++ IDE window showing the execution of a C++ program. The code is in EXP-13.cpp. The program prompts the user to enter a string, which is then reversed and printed.

```

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 }

```

Output window:

```

Enter a string: sai
Reversed string: ias

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

```

Compiler output:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-13.exe
- Output Size: 129,3125 KiB
- Compilation Time: 0.209

```

Line: 27 Col: 2 Sek: 0 Lines: 27 Length: 532 Insert Done parsing in 0.016 seconds

14.

Dev-C++ IDE window showing the execution of a C++ program. The code is in EXP-14.cpp. The program finds the minimum and maximum values in an array and prints the sequences.

```

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 }

```

Output window:

```

Minimum value sequence: 1
Maximum value sequence: 9

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

```

Compiler output:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-14.exe
- Output Size: 127,93140425 KiB
- Compilation Time: 0.198

```

Line: 23 Col: 2 Sek: 0 Lines: 23 Length: 373 Insert Done parsing in 0.015 seconds

15.

The screenshot shows a C++ IDE with the following code in `EXP-15.cpp`:

```

1 #include <stdio.h>
2
3 void strassenMatrixMultiply(int A[3][2], int B[2][2], int C[3][2]) {
4     int M1 = (A[0][0] + A[1][1]) * (B[0][0] + B[1][1]);
5     int M2 = (A[1][0] + A[1][1]) * B[0][0];
6     int M3 = A[0][0] * B[1][1] - B[1][1];
7     int M4 = A[1][1] * (B[0][0] - B[0][1]);
8     int M5 = (A[0][0] + A[0][1]) * B[1][1];
9     int M6 = (A[1][0] - A[0][0]) * B[0][1] + B[0][1];
10    int M7 = (A[0][1] - A[1][1]) * B[1][0] + B[1][1];
11
12    C[0][0] = M1 + M4 - M5 + M7;
13    C[0][1] = M3 + M5;
14    C[1][0] = M2 + M4;
15    C[1][1] = M1 - M2 + M3 + M6;
16 }
17
18 int main() {
19     int A[3][2] = {{1, 2}, {3, 4}};
20     int B[2][2] = {{5, 6}, {7, 8}};
21     int C[3][2];
22
23     strassenMatrixMultiply(A, B, C);
24
25     printf("Resultant Matrix after Strassen's Matrix Multiplication:\n");
26     for (int i = 0; i < 3; i++) {
27         for (int j = 0; j < 2; j++) {
28             printf("%d ", C[i][j]);
29         }
30         printf("\n");
31     }
32
33     return 0;
34 }

```

The output window shows the following text:

```

C:\Users\nagah\Documents\DA\EXP-15.cpp
Resultant Matrix after Strassen's Matrix Multiplication:
19 22
43 50

-----
Process exited after 1.369 seconds with return value 0
Press any key to continue . . .

```

Compiler output:

```

-----
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagah\Documents\DA\EXP-15.exe
Output Size: 129,827,149,4375 KiB
Compilation Time: 0.24s

```

Line: 34 Col: 2 Sel: 0 Lines: 34 Length: 958 Insert Done parsing in 0 seconds

16.

The screenshot shows a C++ IDE with the following code in `EXP-16.cpp`:

```

1 #include <stdio.h>
2
3 void mergeSort(int arr[], int left, int mid, int right) {
4     if (left < right) {
5         int mid = (left + right) / 2;
6         mergeSort(arr, left, mid, right);
7         mergeSort(arr, mid + 1, right, right);
8         merge(arr, left, mid, right);
9     }
10 }
11
12 void merge(int arr[], int left, int mid, int right) {
13     int n1 = mid - left + 1;
14     int n2 = right - mid;
15     int *arr1 = new int[n1];
16     int *arr2 = new int[n2];
17     for (int i = 0; i < n1; i++)
18         arr1[i] = arr[left + i];
19     for (int j = 0; j < n2; j++)
20         arr2[j] = arr[mid + 1 + j];
21     int i = 0, j = 0, k = left;
22     while (i < n1 && j < n2) {
23         if (arr1[i] < arr2[j])
24             arr[k] = arr1[i++];
25         else
26             arr[k] = arr2[j++];
27     }
28     while (i < n1)
29         arr[k] = arr1[i++];
30     while (j < n2)
31         arr[k] = arr2[j++];
32 }
33
34 void mergeSort(int arr[], int left, int right) {
35     if (left < right) {
36         int mid = (left + right) / 2;
37         mergeSort(arr, left, mid, right);
38         mergeSort(arr, mid + 1, right, right);
39         merge(arr, left, mid, right);
40     }
41 }
42
43 int main() {
44     int arr[] = {12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
45     int n = sizeof(arr) / sizeof(arr[0]);
46     mergeSort(arr, 0, n - 1);
47     printf("Sorted array: ");
48     for (int i = 0; i < n; i++)
49         printf("%d ", arr[i]);
50     printf("\n");
51     return 0;
52 }

```

The output window shows the following text:

```

C:\Users\nagah\Documents\DA\EXP-16.cpp
Sorted array: 5 6 7 11 12 13

-----
Process exited after 1.006 seconds with return value 0
Press any key to continue . . .

```

Compiler output:

```

-----
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagah\Documents\DA\EXP-16.exe
Output Size: 129,168,945,3125 KiB
Compilation Time: 0.17s

```

Line: 66 Col: 2 Sel: 0 Lines: 66 Length: 1252 Insert Done parsing in 0 seconds

17.

```

1 #include <stdio.h>
2 void findMaxMin(int arr[], int low, int high, int *max, int *min) {
3     int mid, maxi, max2, mini, 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        }
13        else {
14            *max = arr[high];
15            *min = arr[low];
16        }
17    }
18    else {
19        mid = (low + high) / 2;
20        findMaxMin(arr, low, mid, &max1, &min1);
21        findMaxMin(arr, mid + 1, high, &max2, &min2);
22        *max = (max1 > max2) ? max1 : max2;
23        *min = (min1 < min2) ? min1 : min2;
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     findMaxMin(arr, 0, n - 1, &max, &min);
31     printf("Maximum value: %d\n", max);
32     printf("Minimum value: %d\n", min);
33     return 0;
34 }

```

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 if (num % i == 0) {
8         return false;
9     }
10    else {
11        return isPrime(num, i + 1);
12    }
13 }
14 void generatePrimes(int n, int i) {
15     if (i <= n) {
16         if (isPrime(i, 1)) {
17             printf("%d is a prime number\n", i);
18         }
19         generatePrimes(n, i + 1);
20     }
21 }
22 int main() {
23     int n;
24     printf("Enter the value of n: ");
25     scanf("%d", &n);
26     printf("Prime numbers up to %d are:\n", n);
27     generatePrimes(n, 2);
28     return 0;
29 }

```

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 knapsack algorithm. The output window displays the maximum value in the knapsack as 13.67. The process exited after 1.161 seconds with a return value of 0.

```

4 // int value;
5 //
6 void knapsackGreedy(int W, struct Item items[], int n) {
7     int i, j;
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 - 1; 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[] = {{16, 60}, {20, 100}, {30, 120}};
36     int n = sizeof(items) / sizeof(items[0]);
37     knapsackGreedy(W, items, n);
38     return 0;
39 }

```

Compiler Output:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-19.exe
- Output Size: 129,975,559,975 KLB
- Compilation Time: 0.34s

```

Line: 37 Col: 46 Sel: 0 Lines: 42 Length: 1105 Insert Done parsing in 0.015 seconds

20.

The screenshot shows a C++ IDE with a project named 'EXP-20.cpp'. The code implements a graph algorithm. The output window displays the edge weights and the process exited after 1.093 seconds with a return value of 0.

```

4 // int value;
5 //
6 void knapsackGreedy(int W, struct Item items[], int n) {
7     int i, j;
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 - 1; 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[] = {{16, 60}, {20, 100}, {30, 120}};
36     int n = sizeof(items) / sizeof(items[0]);
37     knapsackGreedy(W, items, n);
38     return 0;
39 }

```

Compiler Output:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-20.exe
- Output Size: 129,201,178,75 KLB
- Compilation Time: 0.23s

```

Line: 5 Col: 13 Sel: 0 Lines: 67 Length: 1372 Insert Done parsing in 0 seconds

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, 9, 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 + 1][k + 1];
5     int i, j;
6
7     for (i = 0; i <= n; i++) {
8         for (j = 0; j <= k; j++) {
9             if (j == 0 || j == i)
10                 c[i][j] = 1;
11             else
12                 c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
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.

The screenshot shows a C++ IDE with a project named 'EXP-24'. The code in 'EXP-10.cpp' is as follows:

```

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 }

```

The output window shows the following text:

```

Enter an integer: 12345
Reversed number = 54321
-----
Process exited after 4.859 seconds with return value 0
Press any key to continue . . .

```

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

24.

The screenshot shows a C++ IDE with a project named 'EXP-25'. The code in 'EXP-10.cpp' is as follows:

```

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 }

```

The output window shows the following text:

```

28 is a perfect number.
-----
Process exited after 1.912 seconds with return value 0
Press any key to continue . . .

```

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

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     dp[pos][mask] = minCost;
22
23     return dp[pos][mask];
24 }
25
26 int main() {
27     int graph[V][V] = {
28         {0, 10, 15, 20},
29         {10, 0, 35, 25},
30         {15, 35, 0, 30},
31         {20, 25, 30, 0}
32     };
33
34     int dp[V][1 << V];
35     for (int i = 0; i < V; i++) {
36         for (int j = 0; j < (1 << V); j++) {
37             dp[i][j] = -1;
38         }
39     }
40
41     int minCost = tsp(graph, 1, 0, dp);
42     printf("Minimum cost for the Travelling Salesman Problem is: %d\n", minCost);
43     return 0;
44 }

```

Minimum cost for the Travelling Salesman Problem is: 80

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

Compiler: TDM-GCC 4.9.2 64-bit Release
Warnings: 0
Output Filename: C:\Users\nagal\Documents\DA\EXP-26.exe
Output Size: 120,96075 KiB
Compilation Time: 0.20s

Line: 42 Col: 5 Sel: 0 Lines: 44 Length: 1032 Insert Done parsing in 0.015 seconds

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 row = 4;
16     printPattern(row);
17     return 0;
18 }

```

1
12
123
1234

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

Compiler: TDM-GCC 4.9.2 64-bit Release
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagal\Documents\DA\EXP-27.exe
Output Size: 120,630671075 KiB
Compilation Time: 0.19s

Line: 15 Col: 18 Sel: 0 Lines: 18 Length: 288 Insert Done parsing in 0.016 seconds

27.

The screenshot shows a C++ IDE with a file named EXP-29.cpp. The code implements the Floyd-Warshall algorithm to find the shortest distances between every pair of vertices in a weighted undirected graph. The graph has 10 vertices (0-9) and the following edges: (0,1) weight 5, (0,2) weight 8, (0,3) weight 9, (1,4) weight 3, (1,5) weight 4, (2,6) weight 1, (3,7) weight 0, (4,8) weight 1, (5,9) weight 0. The output window displays the shortest distances between every pair of vertices:

```

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

Process exited after 0.7291 seconds with return value 0
Press any key to continue . . .

```

28.

The screenshot shows a C++ IDE with a file named EXP-30.cpp. The code implements a program to print Pascal's triangle for a given number of rows. The user entered 4 rows. The output window displays the Pascal's triangle for 4 rows:

```

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

Process exited after 6.434 seconds with return value 0
Press any key to continue . . .

```

29.

```

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

```

Compiler Output:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagal\Documents\DA\EXP-29.exe
- Output Size: 128.68140625 KIB
- Compilation Time: 0.19s

```

Terminal Output:

```

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

```

30.

```

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 }

```

Compiler Output:

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagal\Documents\DA\EXP-31.exe
- Output Size: 128.4658203125 KIB
- Compilation Time: 0.22s

```

Terminal 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 <limits.h>
2 #include <stdio.h>
3
4 void findMinimumMaximum(int arr[], int n)
5 {
6     int i;
7     int min = INT_MIN, max = INT_MAX;
8
9     for (i = 0; i < n; i++) {
10         if (arr[i] < min) {
11             min = arr[i];
12         }
13         if (arr[i] > max) {
14             max = arr[i];
15         }
16     }
17
18     printf("The minimum element is %d", min);
19     printf("\n");
20     printf("The maximum element is %d", max);
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 }

```

Compiler Resources Compile Log Debug Find Results Close

Output Size: 128.64684375 KiB
Compilation Time: 0.23s

Line: 1 Col: 1 Set: 0 Lines: 39 Length: 538 Insert Done parsing in 0.016 seconds

C:\Users\naga\Documents\ID x +

```

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>
2 #include <vector>
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, j;
17     for (i = 0; i < row; i++) {
18         if (board[i][col]) {
19             return false;
20         }
21     }
22     for (i = row, j = col; i >= 0 && j <= N-1; i--, j--) {
23         if (board[i][j]) {
24             return false;
25         }
26     }
27     for (i = row, j = col; i <= N-1 && j <= N-1; i++, j--) {
28         if (board[i][j]) {
29             return false;
30         }
31     }
32     return true;
33}
34
35 bool solveQueensUtil(int board[N][N], int col) {
36     if (col >= N) {
37         return true;
38     }
39     for (int i = 0; i < N; i++) {
40         if (isSafe(board, i, col)) {
41             board[i][col] = 1;
42             if (solveQueensUtil(board, col + 1)) {
43                 return true;
44             }
45         }
46     }
47     return false;
48}
49
50 bool solveQueens() {
51     int board[N][N] = {{0, 0, 0, 0},
52                        {0, 0, 0, 0},
53                        {0, 0, 0, 0},
54                        {0, 0, 0, 0}};
55     if (solveQueensUtil(board, 0) == false) {
56         printf("Solution does not exist");
57         return false;
58     }
59     printSolution(board);
60     return true;
61}
62
63 int main() {
64     solveQueens();
65     return 0;
66}

```

Compiler Resources Compile Log Debug Find Results Close

Output Size: 129.2568359375 KiB
Compilation Time: 0.19s

Line: 1 Col: 1 Set: 0 Lines: 61 Length: 1396 Insert Done parsing in 0 seconds

C:\Users\naga\Documents\ID x +

```

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 AStyle Window Help

Project Classes Debug EXP-33.cpp

```

1 #include <iostream.h>
2
3 int main()
4 {
5     int arr[100] = { 0 };
6     int i, n, sum, m = 34;
7     for (i = 0; i < m; i++)
8         arr[i] = i + 1;
9
10    for (i = 0; i < m; i++)
11    {
12        printf("%d ", arr[i]);
13        if (i % 10 == 9)
14            printf("\n");
15        sum = 0;
16        for (j = 0; j < m; j++)
17            sum += arr[j];
18        printf("Sum = %d\n", sum);
19        if (i % 10 == 9)
20            printf("\n");
21    }
22
23    if (isSubsetSum(arr, m, sum))
24        printf("Found a subset with given sum\n");
25    else
26        printf("No subset with given sum\n");
27
28    return 0;
29 }

```

Compiler Resources Compile Log Debug Find Results Close

Output Size: 128.103515625 KiB
Compilation Time: 0.19s

Shorten compiler paths

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

C:\Users\nagaf\Documents\ID

```

1 2 3 4 5 6 7 8 9 10
1 2 3 4 50 5 6 7 8 9 10
-----
Process exited after 1.016 seconds with return value 0
Press any key to continue . . .

```

34.

File Edit Search View Project Execute Tools AStyle Window Help

Project Classes Debug EXP-34.cpp

```

1 #include <iostream.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[] = { 3, 34, 4, 12, 5, 2 };
11    int sum = 0;
12    int n = sizeof(set) / sizeof(set[0]);
13    if (isSubsetSum(set, n, sum)) printf("Found a subset with given sum\n");
14    else printf("No subset with given sum\n");
15    return 0;
16 }

```

Compiler Resources Compile Log Debug Find Results Close

Output Size: 128.103515625 KiB
Compilation Time: 0.19s

Shorten compiler paths

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

C:\Users\nagaf\Documents\ID

```

Found a subset with given sum
-----
Process exited after 0.6903 seconds with return value 0
Press any key to continue . . .

```

35.

```

1 // Hamiltonian path in a graph
2 #include <iostream>
3 using namespace std;
4
5 int n, m;
6
7 void printSolution(int color[]) {
8     for (int i = 1; i <= n; i++)
9         if (color[i] != 0)
10             return false;
11     return true;
12 }
13
14 bool graphColoringUtil(int graph[][N], int v, int color[]) {
15     if (v == N)
16         return true;
17     for (int i = 1; i <= N; i++) {
18         if (isSafe(i, graph, color, v)) {
19             color[i] = i;
20             if (graphColoringUtil(graph, v + 1, color))
21                 return true;
22             color[i] = 0;
23         }
24     }
25     return false;
26 }
27
28 bool graphColoring(int graph[][N], int v) {
29     int color[N];
30     for (int i = 1; i <= N; i++)
31         color[i] = 0;
32     if (graphColoringUtil(graph, v, color) == false) {
33         printf("Solution does not exist\n");
34         return false;
35     }
36     printSolution(color);
37     return true;
38 }
39
40 void printSolution(int color[]) {
41     printf("Solution exists:\n");
42     printf("Following are the assigned colors:\n");
43     for (int i = 1; i <= N; i++)
44         printf("%d ", color[i]);
45     printf("\n");
46 }
47
48 int main() {
49     int graph[N][N] = {
50         {0, 1, 1, 1},
51         {1, 0, 1, 1},
52         {1, 1, 0, 1},
53         {1, 1, 1, 0}
54     };
55     int v = 1;
56     graphColoring(graph, v);
57     return 0;
58 }

```

Line: 1 Col: 1 Set: 0 Lines: 78 Length: 1245 Insert Done parsing in 0.015 seconds

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.

```

1 // Hamiltonian path in a graph
2 #include <iostream>
3 using namespace std;
4
5 int n, m;
6
7 void printSolution(int color[]) {
8     for (int i = 1; i <= n; i++)
9         if (color[i] != 0)
10             return false;
11     return true;
12 }
13
14 bool graphColoringUtil(int graph[][N], int v, int color[]) {
15     if (v == N)
16         return true;
17     for (int i = 1; i <= N; i++) {
18         if (isSafe(i, graph, color, v)) {
19             color[i] = i;
20             if (graphColoringUtil(graph, v + 1, color))
21                 return true;
22             color[i] = 0;
23         }
24     }
25     return false;
26 }
27
28 bool graphColoring(int graph[][N], int v) {
29     int color[N];
30     for (int i = 1; i <= N; i++)
31         color[i] = 0;
32     if (graphColoringUtil(graph, v, color) == false) {
33         printf("Solution does not exist\n");
34         return false;
35     }
36     printSolution(color);
37     return true;
38 }
39
40 void printSolution(int color[]) {
41     printf("Solution exists:\n");
42     printf("Following are the assigned colors:\n");
43     for (int i = 1; i <= N; i++)
44         printf("%d ", color[i]);
45     printf("\n");
46 }
47
48 int main() {
49     int graph[N][N] = {
50         {0, 1, 1, 1},
51         {1, 0, 1, 1},
52         {1, 1, 0, 1},
53         {1, 1, 1, 0}
54     };
55     int v = 1;
56     graphColoring(graph, v);
57     return 0;
58 }

```

Line: 46 Col: 2 Set: 0 Lines: 46 Length: 1013 Insert Done parsing in 0 seconds

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:
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:

```

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 . . .

```

38.

```

1 #include <stdio.h>
2 #include <limits.h>
3
4 #define N 4
5
6 int cost[h][k] = {
7     {10, 2, 6, 5},
8     {3, 15, 7, 12},
9     {9, 9, 4, 1},
10    {4, 7, 2, 10}
11 };
12
13 int min_cost = INT_MAX;
14 int assigned[h];
15 int visited[h] = {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 < h; i++) {
22                 assigned[i] = visited[i];
23             }
24             return;
25         }
26     }
27     for (int task = 0; task < k; 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 < h; i++) {
41         printf("Worker %d -> Task %d, ", i + 1, assigned[i] + 1);
42     }
43     printf("\n");
44     return 0;
45 }

```

Output:

```

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 . . .

```

39.

```

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 }

```

Output: Element found at index: 2

Process exited after 0.7782 seconds with return value 0
Press any key to continue . . .

40.

```

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 }

```

Output: List after insertion: 15 -> 10 -> 5 -> NULL

Process exited after 1.872 seconds with return value 0
Press any key to continue . . .