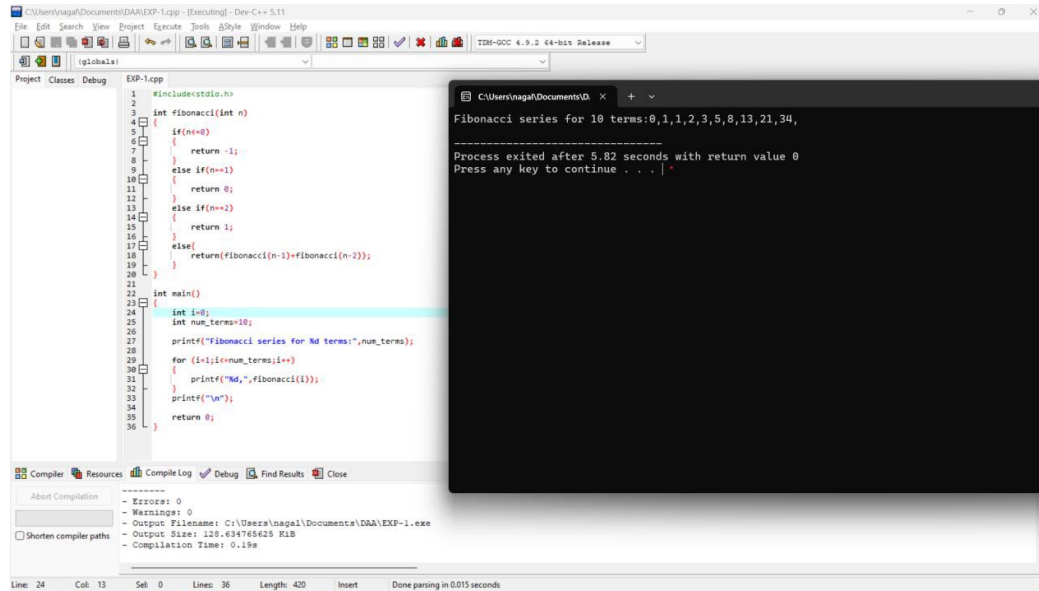


CSA0653-DESIGN AND ANALYSIS OF ALGORITHMS

1.



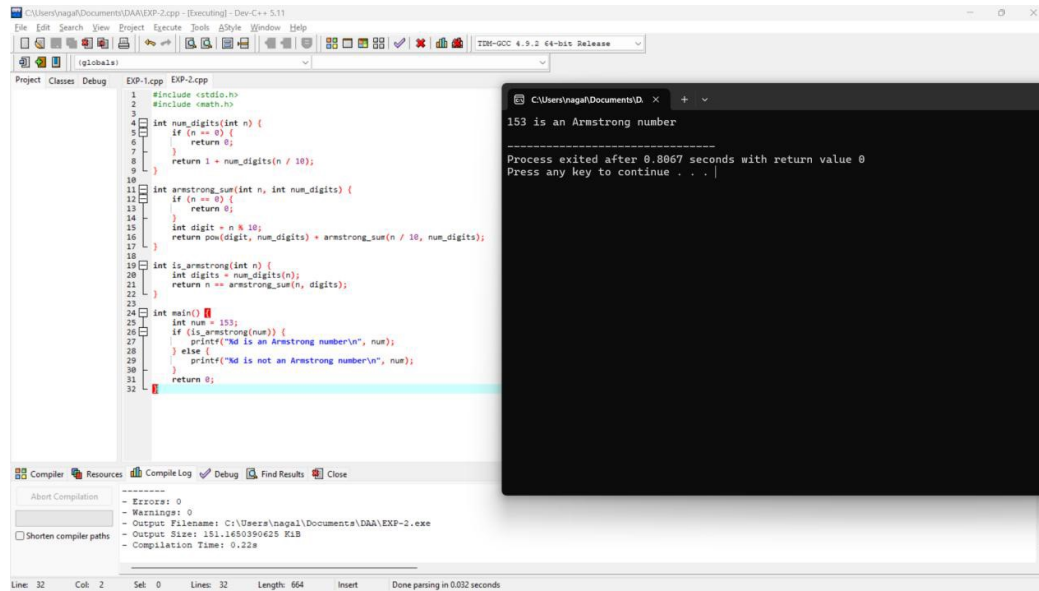
```
1 #include <stdio.h>
2
3 int fibonacci(int n)
4 {
5     if(n==0)
6     {
7         return -1;
8     }
9     else if(n==1)
10    {
11        return 0;
12    }
13    else if(n==2)
14    {
15        return 1;
16    }
17    else
18    {
19        return(fibonacci(n-1)+fibonacci(n-2));
20    }
21 }
22
23 int main()
24 {
25     int i=0;
26     int num_terms=10;
27     printf("Fibonacci series for %d terms:", num_terms);
28     for (i=1;i<=num_terms;i++)
29     {
30         printf("%d,", fibonacci(i));
31     }
32     printf("\n");
33     return 0;
34 }
```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-1.exe
- Output Size: 120.634765625 KiB
- Compilation Time: 0.19s

Line: 24 Col: 13 Sel: 0 Lines: 36 Length: 420 Insert Done parsing in 0.015 seconds

C:\Users\nagaj\Documents\DA\EXP-1.exe
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.



```
1 #include <stdio.h>
2 #include <math.h>
3
4 int num_digits(int n) {
5     if (n == 0) {
6         return 0;
7     }
8     return 1 + num_digits(n / 10);
9 }
10
11 int armstrong_sum(int n, int num_digits) {
12     if (n == 0) {
13         return 0;
14     }
15     int digit = n % 10;
16     return pow(digit, num_digits) + armstrong_sum(n / 10, num_digits);
17 }
18
19 int is_armstrong(int n) {
20     int digits = num_digits(n);
21     return n == armstrong_sum(n, digits);
22 }
23
24 int main() {
25     int num = 153;
26     if (is_armstrong(num)) {
27         printf("%d is an Armstrong number\n", num);
28     } else {
29         printf("%d is not an Armstrong number\n", num);
30     }
31     return 0;
32 }
```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagaj\Documents\DA\EXP-2.exe
- Output Size: 151.1650390625 KiB
- Compilation Time: 0.22s

Line: 32 Col: 2 Sel: 0 Lines: 32 Length: 664 Insert Done parsing in 0.032 seconds

C:\Users\nagaj\Documents\DA\EXP-2.exe
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 a project named 'EXP-3.cpp'. The code defines a recursive function `gcd` and a `main` function that calls it with `num1=20` and `num2=40`. The output window displays the result: 'GCD of 20 & 40 is 20'. The compiler log shows no errors or warnings.

```
#include <stdio.h>

int gcd(int a, int b)
{
    if(b==0){
        return a;
    }
    else{
        return(b, a%b);
    }
}

int main()
{
    int num1=20;
    int num2=40;

    printf("GCD of %d & %d is %d", num1, num2, gcd(num1, num2));

    return 0;
}
```

Output: GCD of 20 & 40 is 20

Process exited after 2.416 seconds with return value 0

Press any key to continue . . .

4.

The screenshot shows a C++ IDE with a project named 'EXP-4.cpp'. The code defines a recursive function `find_largest` and a `main` function that calls it with an array `{3, 5, 7, 2, 8, 1}`. The output window displays the result: 'The largest element in the array is 8'. The compiler log shows no errors or warnings.

```
#include <stdio.h>

int find_largest(int arr[], int n)
{
    if (n == 1) {
        return arr[0];
    }

    int max_of_rest = find_largest(arr, n - 1);
    return (arr[n - 1] > max_of_rest) ? arr[n - 1] : max_of_rest;
}

int main()
{
    int arr[] = {3, 5, 7, 2, 8, 1};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("The largest element in the array is %d\n", find_largest(arr, n));

    return 0;
}
```

Output: The largest element in the array is 8

Process exited after 1.81 seconds with return value 0

Press any key to continue . . .

5.

The screenshot shows a C++ IDE with a project named 'EXP-5.cpp'. The code is as follows:

```

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 }

```

The output window shows the following text:

```

C:\Users\nagah\Documents\EXP-5.cpp
Factorial of 5 is 120
Process exited after 0.6644 seconds with return value 0
Press any key to continue . . .

```

The compiler window shows the following output:

```

Compiler Resources Compile Log Debug Find Results Close
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagah\Documents\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 is as follows:

```

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

```

The output window shows the following text:

```

C:\Users\nagah\Documents\EXP-6.cpp
Enter the number:2
2 is a prime number...
Process exited after 7.386 seconds with return value 0
Press any key to continue . . .

```

The compiler window shows the following output:

```

Compiler Resources Compile Log Debug Find Results Close
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\nagah\Documents\EXP-6.exe
- Output Size: 128.4015425 KIB
- Compilation Time: 0.20s

```

7.

```

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 int main() {
18     int arr[] = {64, 25, 12, 22, 11, 64};
19     int n = sizeof(arr)/sizeof(arr[0]);
20     selection_sort(arr, n);
21     printf("Sorted array: ");
22     for (int i = 0; i < n; i++) {
23         printf("%d ", arr[i]);
24     }
25     printf("\n");
26     return 0;
27 }

```

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
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagah\Documents\DA\EXP-7.exe
Output Size: 128.642576125 KIB
Compilation Time: 0.23s

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

8.

```

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, 64, 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
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagah\Documents\DA\EXP-8.exe
Output Size: 128.6396464375 KIB
Compilation Time: 0.17s

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

9.

```

1 //Matrix multiplication
2 #include <iostream>
3 #include <vector>
4 #include <string>
5 using namespace std;
6
7 void printMatrix(vector<vector<int>>& mat, int row, int col) {
8     for (int i = 0; i < row; i++) {
9         for (int j = 0; j < col; j++) {
10             cout << mat[i][j] << " ";
11         }
12         cout << endl;
13     }
14 }
15
16 void printMatrix(vector<vector<int>>& mat, int row, int col) {
17     for (int i = 0; i < row; i++) {
18         for (int j = 0; j < col; j++) {
19             cout << mat[i][j] << " ";
20         }
21         cout << endl;
22     }
23 }
24
25 void multiplyMatrix(vector<vector<int>>& mat1, vector<vector<int>>& mat2, vector<vector<int>>& result, int row1, int col1, int col2) {
26     for (int i = 0; i < row1; i++) {
27         for (int j = 0; j < col2; j++) {
28             result[i][j] = 0;
29             for (int k = 0; k < col1; k++) {
30                 result[i][j] += mat1[i][k] * mat2[k][j];
31             }
32         }
33     }
34 }
35
36 int main() {
37     int row1, col1, row2, col2;
38     cout << "Enter rows and columns for the first matrix: ";
39     cin >> row1 >> col1;
40     cout << "Enter rows and columns for the second matrix: ";
41     cin >> row2 >> col2;
42
43     if (col1 != row2) {
44         cout << "Error: Number of columns in the first matrix must be equal to the number of rows in the second matrix.\n";
45         return 1;
46     }
47
48     vector<vector<int>> mat1(row1, vector<int>(col1));
49     vector<vector<int>> mat2(row2, vector<int>(col2));
50     vector<vector<int>> result(row1, vector<int>(col2));
51
52     cout << "Enter the first matrix:\n";
53     for (int i = 0; i < row1; i++) {
54         for (int j = 0; j < col1; j++) {
55             cin >> mat1[i][j];
56         }
57     }
58
59     cout << "Enter the second matrix:\n";
60     for (int i = 0; i < row2; i++) {
61         for (int j = 0; j < col2; j++) {
62             cin >> mat2[i][j];
63         }
64     }
65
66     multiplyMatrix(mat1, mat2, result, row1, col1, col2);
67
68     cout << "Resultant matrix:\n";
69     printMatrix(result, row1, col2);
70
71     return 0;
72 }

```

Output:

```

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 <string.h>
2 #include <stdbool.h>
3 #include <string>
4 using namespace std;
5
6 bool isPalindrome(char *s, int start, int end) {
7     if (start >= end) {
8         return true;
9     }
10    if (s[start] == s[end]) {
11        return isPalindrome(s, start + 1, end - 1);
12    } else {
13        return false;
14    }
15 }
16
17 int main() {
18     char string[] = "racecar";
19     int length = strlen(string);
20     if (isPalindrome(string, 0, length - 1)) {
21         printf("%s is a palindrome\n", string);
22     } else {
23         printf("%s is not a palindrome\n", string);
24     }
25     return 0;
26 }

```

Output:

```

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 file named `EXP-11.cpp`. The code implements a `copyString` function that copies a string from `source` to `destination` starting at a given `index`. The `main` function prompts the user to enter a string, reads it using `gets`, and then prints both the source and copied strings.

```

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     gets(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 }

```

The output window shows the following text:

```

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 . . .

```

The compiler window shows no errors or warnings, and the output file is `C:\Users\nagah\Documents\DA\EXP-11.exe`.

12.

The screenshot shows a C++ IDE with a file named `EXP-12.cpp`. The code implements a `binarySearch` function that searches for a `key` in a sorted array `arr` between `left` and `right` indices. The `main` function defines an array, sets a key, and calls the `binarySearch` function, printing the result.

```

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 }

```

The output window shows the following text:

```

Element found at index 5.

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

```

The compiler window shows no errors or warnings, and the output file is `C:\Users\nagah\Documents\DA\EXP-12.exe`.

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
9     char temp = str[start];
10    str[start] = str[end];
11    str[end] = temp;
12
13    reverseString(str, start + 1, end - 1);
14 }
15
16 int main() {
17     char str[100];
18     printf("Enter a string: ");
19     fgets(str, sizeof(str), stdin);
20     str[strcspn(str, "\n")] = '\0';
21
22     reverseString(str, 0, strlen(str) - 1);
23
24     printf("Reversed string: %s\n", str);
25
26     return 0;
27 }

```

Enter a string: sai
Reversed string: ias

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

Compiler: g++ 4.9.2 64-bit Release
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagah\Documents\DA\EXP-13.exe
Output Size: 129.3125 KiB
Compilation Time: 0.28s

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
22    return 0;
23 }

```

Minimum value sequence: 1
Maximum value sequence: 9

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

Compiler: g++ 4.9.2 64-bit Release
Errors: 0
Warnings: 0
Output Filename: C:\Users\nagah\Documents\DA\EXP-14.exe
Output Size: 127.93146425 KiB
Compilation Time: 0.19s

15.

The screenshot shows a C++ IDE with a project named 'DAA'. The active file is 'EXP-15.cpp', which implements Strassen's Matrix Multiplication algorithm. The code defines a function `strassenMatrixMultiply` that takes two 2x2 matrices `A` and `B` and returns their product `C`. The `main` function initializes matrix `A` as $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and matrix `B` as $\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$, then calls `strassenMatrixMultiply(A, B, C)` and prints the resulting matrix `C`.

```

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

```

The output window shows the following text:

```

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 . . .

```

The IDE status bar at the bottom indicates: Line: 34, Col: 2, Sel: 0, Lines: 34, Length: 958, Insert, Done parsing in 0 seconds.

16.

The screenshot shows a C++ IDE with a project named 'DAA'. The active file is 'EXP-16.cpp', which implements the Merge Sort algorithm. The code defines a function `mergeSort` that recursively sorts an array. The `main` function initializes an array `arr` with the values `{5, 6, 7, 11, 12, 13}` and calls `mergeSort(arr, 0, arr.length - 1)`.

```

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

```

The output window shows the following text:

```

Sorted array: 5 6 7 11 12 13

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

```

The IDE status bar at the bottom indicates: 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     if (low == high) {
4         *max = arr[low];
5         *min = arr[low];
6     }
7     else if (high - low == 1) {
8         if (arr[low] > arr[high]) {
9             *max = arr[low];
10            *min = arr[high];
11        }
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        }
11        else {
12            return isPrime(num, i - 1);
13        }
14    }
15 }
16 void generatePrimes(int n, int i) {
17     if (i <= n) {
18         if (isPrime(i, i / 2)) {
19             printf("%d is a prime number\n", i);
20             generatePrimes(n, i + 1);
21         }
22     }
23 }
24
25 int main() {
26     int n;
27
28     printf("Enter the value of n: ");
29     scanf("%d", &n);
30
31     printf("Prime numbers up to %d are:\n", n);
32     generatePrimes(n, 1);
33
34     return 0;
35 }

```

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 . . .