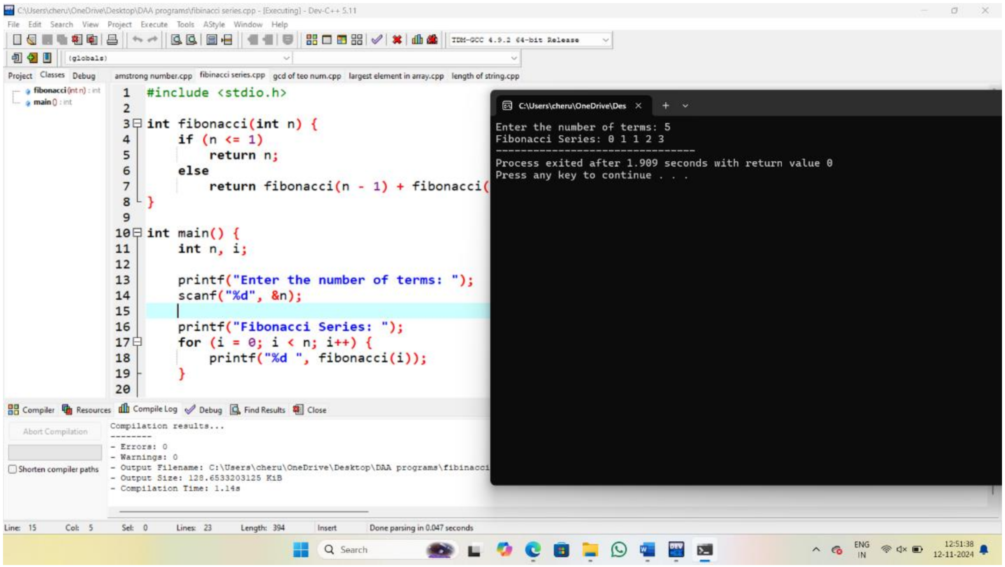
1.Write a program to Print Fibonacci Series using recursion.

Program:

```
#include <stdio.h>
int fibonacci(int n) {
   if (n <= 1)
      return n;
   else
      return fibonacci(n - 1) + fibonacci(n - 2);
}
int main() {
   int n, i;
  printf("Enter the number of terms: ");
   scanf("%d", &n);
   printf("Fibonacci Series: ");
   for (i = 0; i < n; i++) {
      printf("%d ", fibonacci(i));
   }
   return 0;
}
```

Output:


C:\Users\cheru\OneDrive\Desktop\DAA programs\fibinacci series.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
amstrong number.cpp
fibinacci series.cpp
gcd of teo num.cpp
largest element in array.cpp
length of string.cpp
fibonacci (int n) : int
main () : int
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 15 Col: 5 Sel: 0 Lines: 23 Length: 394 Insert Done parsing in 0.047 seconds
C:\Users\cheru\OneDrive\Des
Search
ENG IN
12:51:38
12-11-2024

```
#include <stdio.h>

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

int main() {
    int n, i;

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

    printf("Fibonacci Series: ");
    for (i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }

```

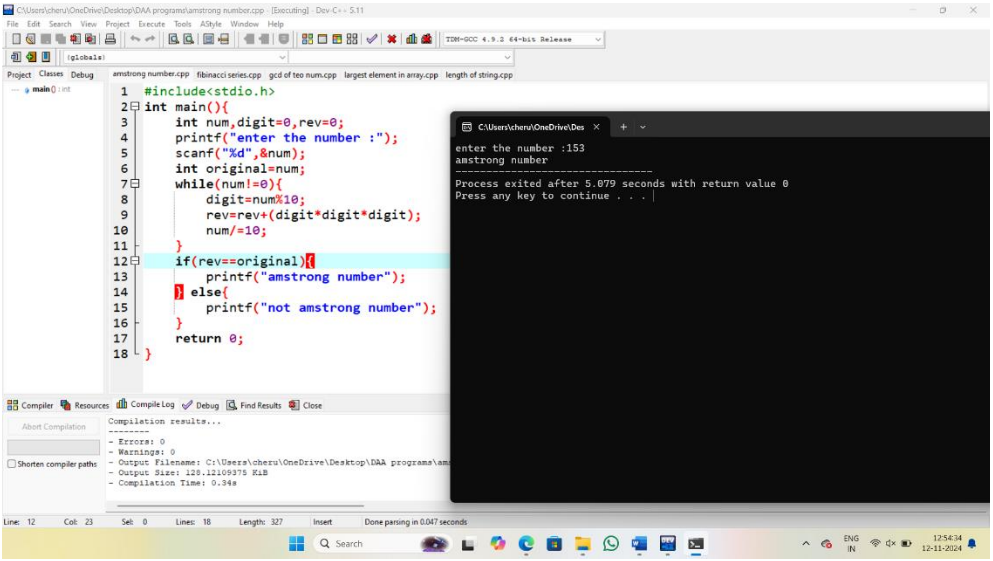
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\fibinacci
- Output Size: 128.6533203125 KiB
- Compilation Time: 1.14s
```

```
Enter the number of terms: 5
Fibonacci Series: 0 1 1 2 3
--------------------------------
Process exited after 1.909 seconds with return value 0
Press any key to continue . . .
```

2.Write a program to check the given no is Armstrong or not.

```
#include<stdio.h>
int main(){
        int num,digit=0,rev=0;
        printf("enter the number :");
        scanf("%d",&num);
        int original=num;
        while(num!=0){
                digit=num%10;
                rev=rev+(digit*digit*digit);
                num/=10;
        }
        if(rev==original){
                printf("amstrong number");
        } else{
                printf("not amstrong number");
        }
        return 0;
}
```

Output:



C:\Users\cheru\OneDrive\Desktop\DAA programs\amstrong number.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
main () : int
amstrong number.cpp
fibinacci series.cpp
gcd of teo num.cpp
largest element in array.cpp
length of string.cpp
C:\Users\cheru\OneDrive\Des
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 12 Col: 23 Sel: 0 Lines: 18 Length: 327 Insert Done parsing in 0.047 seconds
Search
ENG IN
12:54:34 12-11-2024

```
#include<stdio.h>
int main(){
    int num,digit=0,rev=0;
    printf("enter the number :");
    scanf("%d",&num);
    int original=num;
    while(num!=0){
        digit=num%10;
        rev=rev+(digit*digit*digit);
        num/=10;
    }
    if(rev==original){
        printf("amstrong number");
    } else{
        printf("not amstrong number");
    }
    return 0;
}
```

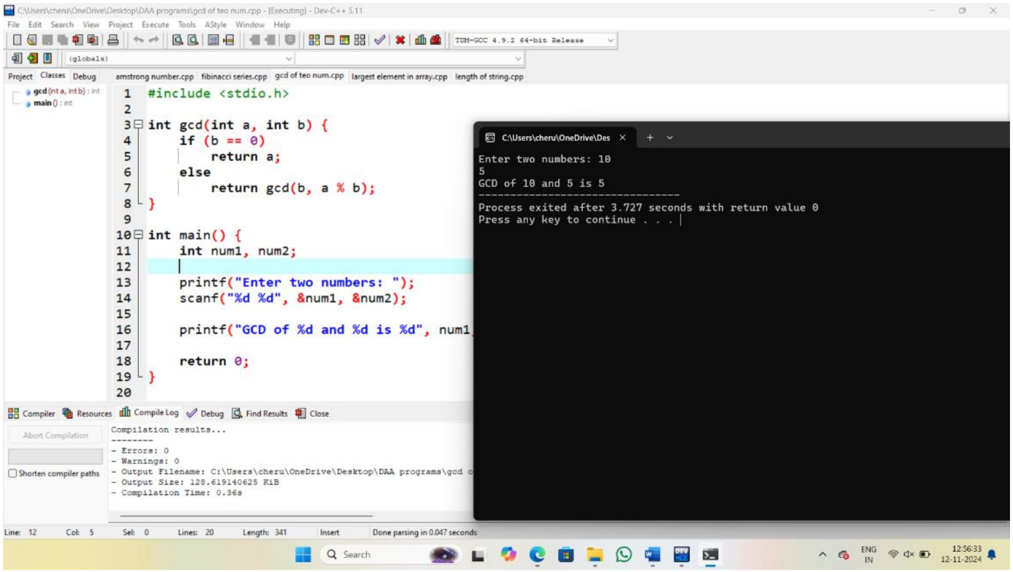
```
enter the number :153
amstrong number
--------------------------------
Process exited after 5.079 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\am
- Output Size: 128.12109375 KiB
- Compilation Time: 0.34s
```

3.Program to find the GCD of two numbers .

```
#include <stdio.h>
int gcd(int a, int b) {
   if (b == 0)
       return a;
   else
       return gcd(b, a % b);
}
int main() {
   int num1, num2;
   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);
   printf("GCD of %d and %d is %d", num1, num2, gcd(num1, num2));
   return 0;
}
```

Output:


C:\Users\cheru\OneDrive\Desktop\DAA programs\gcd of teo num.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
amstrong number.cpp fibinacci series.cpp gcd of teo num.cpp largest element in array.cpp length of string.cpp
gcd (int a, int b) : int
main () : int
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 12 Col: 5 Sel: 0 Lines: 20 Length: 341 Insert Done parsing in 0.047 seconds
C:\Users\cheru\OneDrive\Des
Search
ENG IN
12:56:33 12-11-2024

```
#include <stdio.h>

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

int main() {
    int num1, num2;
    
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    printf("GCD of %d and %d is %d", num1

    return 0;
}
```

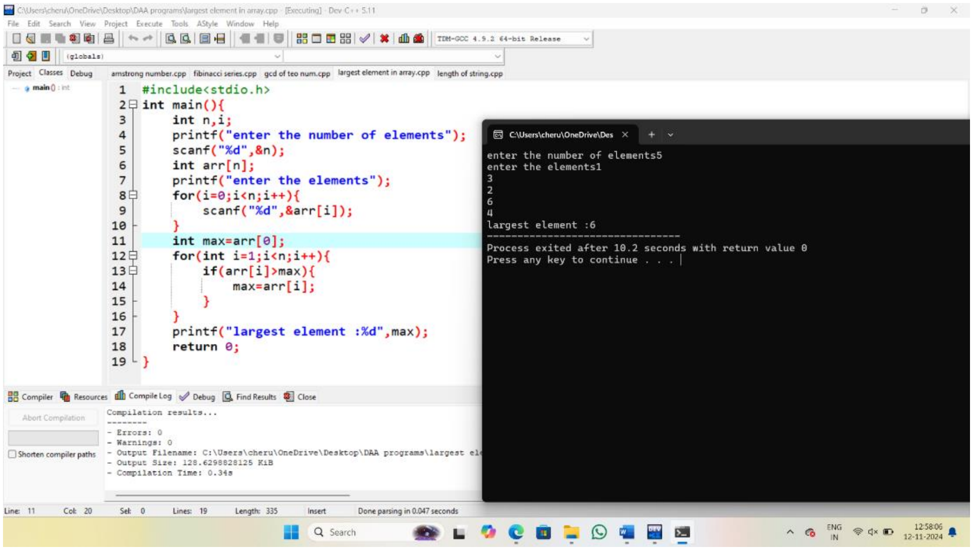
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\gcd o
- Output Size: 128.619140625 KiB
- Compilation Time: 0.36s
```

```
Enter two numbers: 10
5
GCD of 10 and 5 is 5
--------------------------------
Process exited after 3.727 seconds with return value 0
Press any key to continue . . .
```

4. Write a program to get the largest element of an array.

```
#include<stdio.h>

int main(){

        int n,i;

        printf("enter the number of elements");

        scanf("%d",&n);

        int arr[n];

        printf("enter the elements");

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

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

        }

        int max=arr[0];

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

                if(arr[i]>max){
                        max=arr[i];
                }
        }
        printf("largest element :%d",max);
        return 0;
}
```

Output:



C:\Users\cheru\OneDrive\Desktop\DAA programs\largest element in array.cpp - [Executing] - Dev-C++ 5.11

```
#include<stdio.h>
int main(){
	int n,i;
	printf("enter the number of elements");
	scanf("%d",&n);
	int arr[n];
	printf("enter the elements");
	for(i=0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	int max=arr[0];
	for(int i=1;i<n;i++){
		if(arr[i]>max){
			max=arr[i];
		}
	}
	printf("largest element :%d",max);
	return 0;
}
```

```
enter the number of elements5
enter the elements1
3
2
6
4
largest element :6
--------------------------------
Process exited after 10.2 seconds with return value 0
Press any key to continue . . .
```

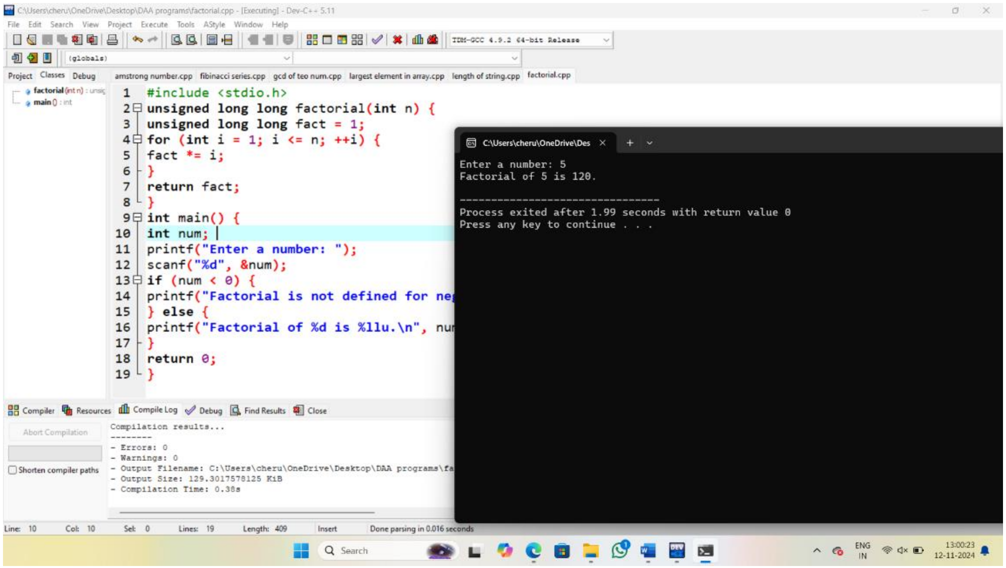
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\largest el
- Output Size: 128.6298828125 KiB
- Compilation Time: 0.34s
```

File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
amstrong number.cpp fibinacci series.cpp gcd of teo num.cpp largest element in array.cpp length of string.cpp
main () : int
C:\Users\cheru\OneDrive\Des
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 11 Col: 20 Sel: 0 Lines: 19 Length: 335 Insert Done parsing in 0.047 seconds
Search
ENG IN
12:58:06 12-11-2024

5. Write a program to find the Factorial of a number.

```
#include <stdio.h>
unsigned long long factorial(int n) {
unsigned long long fact = 1;
for (int i = 1; i <= n; ++i) {
fact *= i;
}
return fact;
}
int main() {
int num;
printf("Enter a number: ");
scanf("%d", &num);
if (num < 0) {
printf("Factorial is not defined for negative numbers.\n");
} else {
printf("Factorial of %d is %llu.\n", num, factorial(num));
}
return 0;
}
```

Output:



C:\Users\cheru\OneDrive\Desktop\DAA programs\factorial.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
amstrong number.cpp fibinacci series.cpp gcd of teo num.cpp largest element in array.cpp length of string.cpp factorial.cpp
factorial (int n) : unsig
main () : int
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 10 Col: 10 Sel: 0 Lines: 19 Length: 409 Insert Done parsing in 0.016 seconds
C:\Users\cheru\OneDrive\Des
Search
ENG IN
13:00:23 12-11-2024

```
#include <stdio.h>
unsigned long long factorial(int n) {
unsigned long long fact = 1;
for (int i = 1; i <= n; ++i) {
fact *= i;
}
return fact;
}
int main() {
int num;
printf("Enter a number: ");
scanf("%d", &num);
if (num < 0) {
printf("Factorial is not defined for ne
} else {
printf("Factorial of %d is %llu.\n", nu
}
return 0;
}
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\fa
- Output Size: 129.3017578125 KiB
- Compilation Time: 0.38s
```

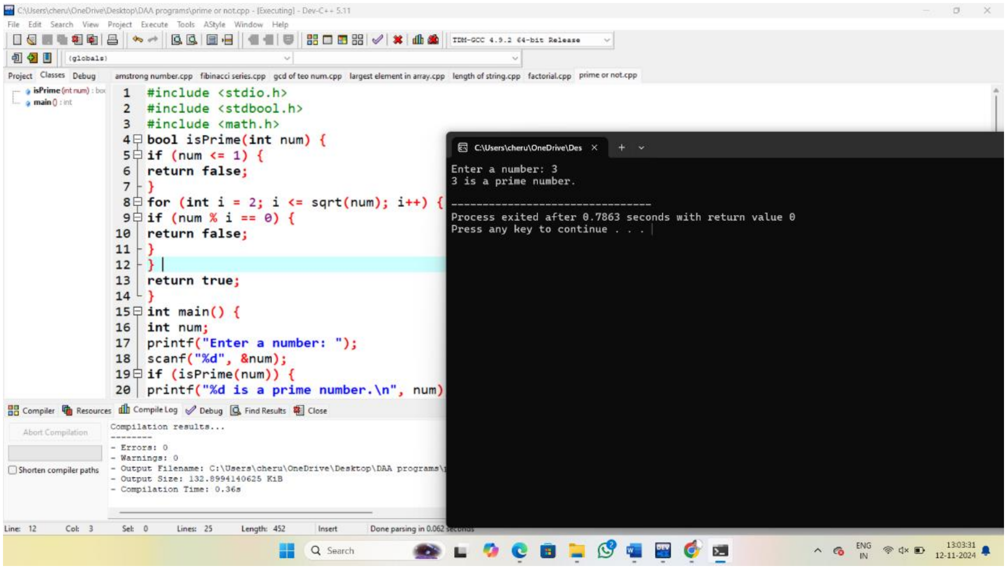
```
Enter a number: 5
Factorial of 5 is 120.

--------------------------------
Process exited after 1.99 seconds with return value 0
Press any key to continue . . .
```

6.Write a program to check a number is a prime number or not .

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
bool isPrime(int num) {
if (num <= 1) {
return false;
}
for (int i = 2; i <= sqrt(num); i++) {
if (num % i == 0) {
```

```
return false;
} }
return true;
}
int main() {
int num;
printf("Enter a number: ");
scanf("%d", &num);
if (isPrime(num)) {
printf("%d is a prime number.\n", num);
} else {
printf("%d is not a prime number.\n", num);
}
return 0;
}
```

Output:


C:\Users\cheru\OneDrive\Desktop\DAA programs\prime or not.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
amstrong number.cpp fibinacci series.cpp gcd of teo num.cpp largest element in array.cpp length of string.cpp factorial.cpp prime or not.cpp
isPrime (int num) : boo
main () : int
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 12 Col: 3 Sel: 0 Lines: 25 Length: 452 Insert Done parsing in 0.062 seconds
C:\Users\cheru\OneDrive\Des
Search
ENG IN
13:03:31
12-11-2024

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
bool isPrime(int num) {
if (num <= 1) {
return false;
}
for (int i = 2; i <= sqrt(num); i++) {
if (num % i == 0) {
return false;
}
}
return true;
}
int main() {
int num;
printf("Enter a number: ");
scanf("%d", &num);
if (isPrime(num)) {
printf("%d is a prime number.\n", num)
```

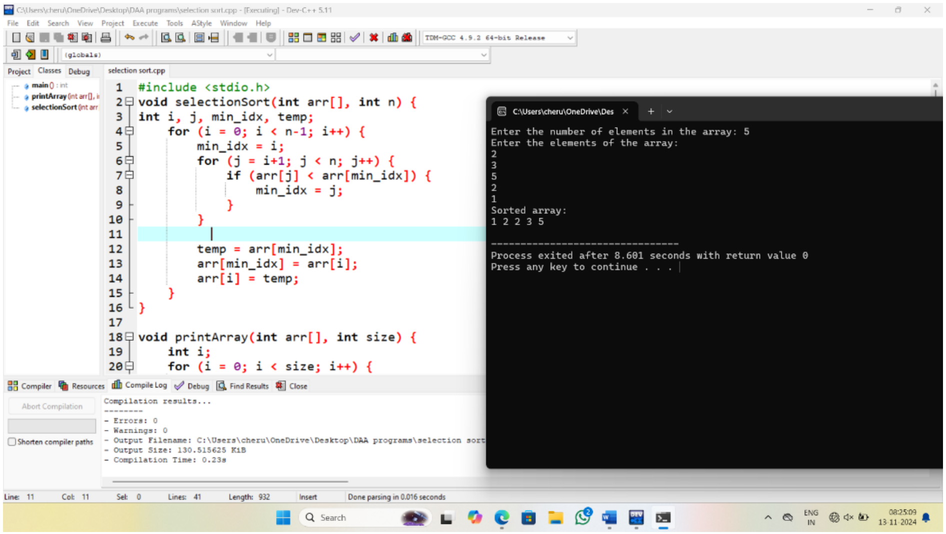
```
Enter a number: 3
3 is a prime number.

--------------------------------
Process exited after 0.7863 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\
- Output Size: 132.8994140625 KiB
- Compilation Time: 0.36s
```

7.Write a program to perform Selection sort.

```
#include <stdio.h>
void selectionSort(int arr[], int n) {
int i, j, min_idx, temp;
    for (i = 0; i < n-1; i++) {
        min_idx = i;
        for (j = i+1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}
void printArray(int arr[], int size) {
    int i;
    for (i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
printf("\n");
}
 int main() {
    int n, i;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements of the array:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
   return 0;
}
```

Output:



C:\Users\cheru\OneDrive\Desktop\DAA programs\selection sort.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
main () : int
printArray (int arr[], i
selectionSort (int arr
selection sort.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 11 Col: 11 Sel: 0 Lines: 41 Length: 932 Insert Done parsing in 0.016 seconds
C:\Users\cheru\OneDrive\Des
Search
ENG IN
08:25:09
13-11-2024

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

        temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

void printArray(int arr[], int size) {
    int i;
    for (i = 0; i < size; i++) {
```

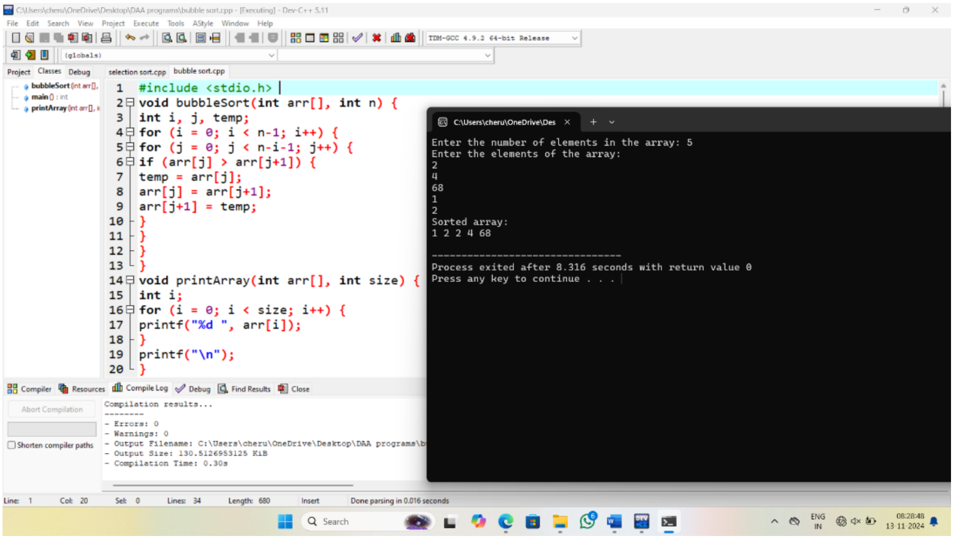
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\selection sort
- Output Size: 130.515625 KiB
- Compilation Time: 0.23s
```

```
Enter the number of elements in the array: 5
Enter the elements of the array:
2
3
5
2
1
Sorted array:
1 2 2 3 5

--------------------------------
Process exited after 8.601 seconds with return value 0
Press any key to continue . . .
```

8.Write a program to perform Bubble sort.

```
#include <stdio.h>
void bubbleSort(int arr[], int n) {
int i, j, temp;
for (i = 0; i < n-1; i++) {
for (j = 0; j < n-i-1; j++) {
if (arr[j] > arr[j+1]) {
temp = arr[j];
arr[j] = arr[j+1];
arr[j+1] = temp;
}
}
}
}
void printArray(int arr[], int size) {
int i;
for (i = 0; i < size; i++) {
printf("%d ", arr[i]);
}
printf("\n");
}
int main() {
int n, i;
printf("Enter the number of elements in the array: ");
scanf("%d", &n);
int arr[n];
printf("Enter the elements of the array:\n");
for (i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
bubbleSort(arr, n);
printf("Sorted array: \n");
printArray(arr, n);
return 0;
}
```

Output:



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

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\b
- Output Size: 130.5126953125 KiB
- Compilation Time: 0.30s
```

```
Enter the number of elements in the array: 5
Enter the elements of the array:
2
4
68
1
2
Sorted array:
1 2 2 4 68

--------------------------------
Process exited after 8.316 seconds with return value 0
Press any key to continue . . .
```

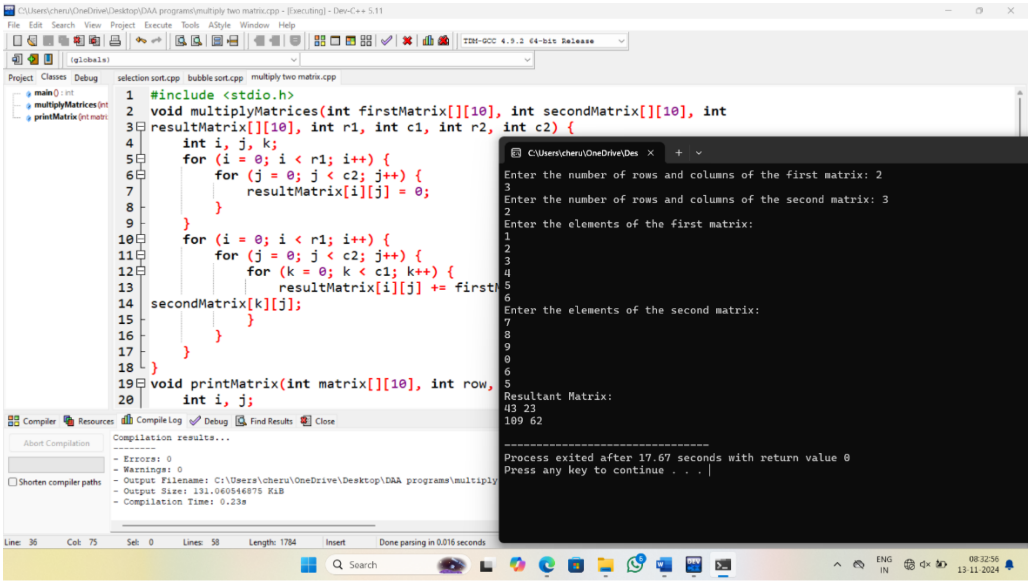
C:\Users\cheru\OneDrive\Desktop\DAA programs\bubble sort.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
selection sort.cpp
bubble sort.cpp
bubbleSort (int arr[], i
main () : int
printArray (int arr[], i
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 1 Col: 20 Sel: 0 Lines: 34 Length: 680 Insert Done parsing in 0.016 seconds
C:\Users\cheru\OneDrive\Des
Search
ENG IN
08:28:48
13-11-2024

9.Write a program for to multiply two Matrix.

```
#include <stdio.h>
void multiplyMatrices(int firstMatrix[][10], int secondMatrix[][10], int
resultMatrix[][10], int r1, int c1, int r2, int c2) {
    int i, j, k;
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c2; j++) {
            resultMatrix[i][j] = 0;
        }
    }
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c2; j++) {
            for (k = 0; k < c1; k++) {
                resultMatrix[i][j] += firstMatrix[i][k] *
secondMatrix[k][j];
            }
        }
    }
}
```

```
void printMatrix(int matrix[][10], int row, int col) {
    int i, j;
    for (i = 0; i < row; i++) {
        for (j = 0; j < col; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int r1, c1, r2, c2, i, j;
    printf("Enter the number of rows and columns of the first matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter the number of rows and columns of the second matrix: ");
    scanf("%d %d", &r2, &c2);
    if (c1 != r2) {
        printf("Error! Column of the first matrix must be equal to row of the second matrix.\n");
        return -1;
    }
    int firstMatrix[10][10], secondMatrix[10][10], resultMatrix[10][10];
    printf("Enter the elements of the first matrix:\n");
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c1; j++) {
            scanf("%d", &firstMatrix[i][j]);
        }
    }
printf("Enter the elements of the second matrix:\n");
for (i = 0; i < r2; i++) {
for (j = 0; j < c2; j++) {
scanf("%d", &secondMatrix[i][j]);
}
}
multiplyMatrices(firstMatrix, secondMatrix, resultMatrix, r1, c1, r2,
c2);
printf("Resultant Matrix:\n");
printMatrix(resultMatrix, r1, c2);
return 0;
}
```

Output:


C:\Users\cheru\OneDrive\Desktop\DAA programs\multiply two matrix.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
selection sort.cpp bubble sort.cpp multiply two matrix.cpp
main () : int
multiplyMatrices (int
printMatrix (int matri
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
C:\Users\cheru\OneDrive\Des
Line: 36 Col: 75 Sel: 0 Lines: 58 Length: 1784 Insert Done parsing in 0.016 seconds
Search
ENG IN
08:32:56
13-11-2024

```
#include <stdio.h>
void multiplyMatrices(int firstMatrix[][10], int secondMatrix[][10], int
resultMatrix[][10], int r1, int c1, int r2, int c2) {
    int i, j, k;
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c2; j++) {
            resultMatrix[i][j] = 0;
        }
    }
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c2; j++) {
            for (k = 0; k < c1; k++) {
                resultMatrix[i][j] += first
secondMatrix[k][j];
            }
        }
    }
}
void printMatrix(int matrix[][10], int row,
    int i, j;
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\multiply
- Output Size: 131.060546875 KiB
- Compilation Time: 0.23s
```

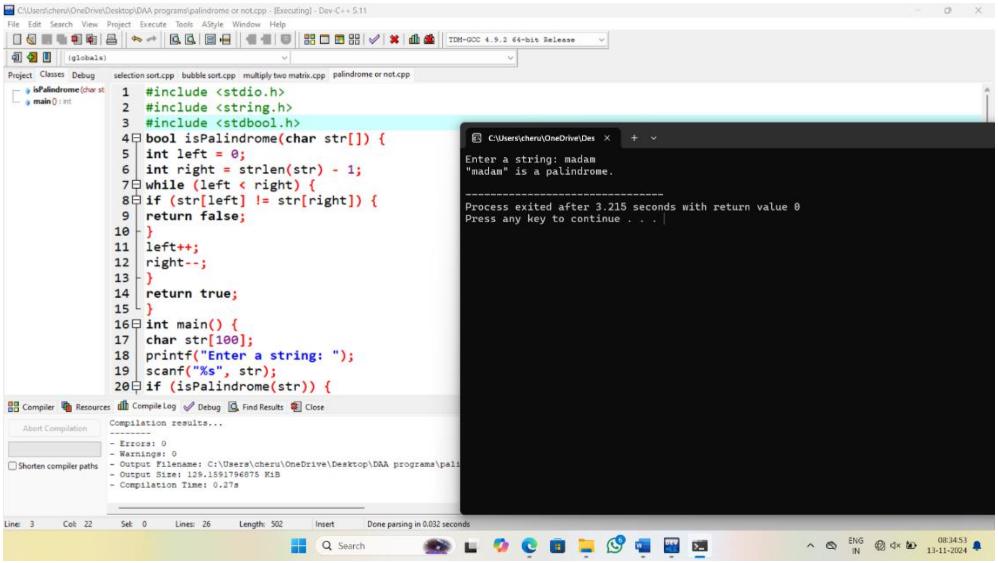
```
Enter the number of rows and columns of the first matrix: 2
3
Enter the number of rows and columns of the second matrix: 3
2
Enter the elements of the first matrix:
1
2
3
4
5
6
Enter the elements of the second matrix:
7
8
9
0
6
5
Resultant Matrix:
43 23
109 62

--------------------------------
Process exited after 17.67 seconds with return value 0
Press any key to continue . . .
```

10.Write a program for to check whether a given String is Palindrome or not.

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

```
bool isPalindrome(char str[]) {
int left = 0;
int right = strlen(str) - 1;
while (left < right) {
if (str[left] != str[right]) {
return false;
}
left++;
right--;
}
return true;
}
int main() {
char str[100];
printf("Enter a string: ");
scanf("%s", str);
if (isPalindrome(str)) {
printf("\"%s\" is a palindrome.\n", str);
} else {
printf("\"%s\" is not a palindrome.\n", str);
}
return 0;
}
```

Output:


File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
selection sort.cpp bubble sort.cpp multiply two matrix.cpp palindrome or not.cpp
isPalindrome (char st
main () : int
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Search

C:\Users\cheru\OneDrive\Desktop\DAA programs\palindrome or not.cpp - [Executing] - Dev-C++ 5.11

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
bool isPalindrome(char str[]) {
int left = 0;
int right = strlen(str) - 1;
while (left < right) {
if (str[left] != str[right]) {
return false;
}
left++;
right--;
}
return true;
}
int main() {
char str[100];
printf("Enter a string: ");
scanf("%s", str);
if (isPalindrome(str)) {
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\pali
- Output Size: 129.1591796875 KiB
- Compilation Time: 0.27s
```

Line: 3 Col: 22 Sel: 0 Lines: 26 Length: 502 Insert Done parsing in 0.032 seconds

C:\Users\cheru\OneDrive\Des

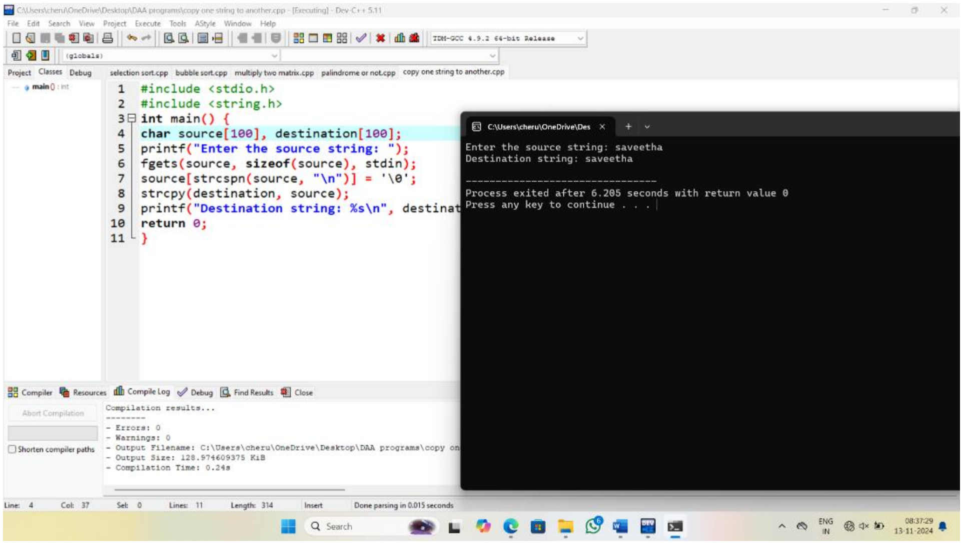
```
Enter a string: madam
"madam" is a palindrome.

--------------------------------
Process exited after 3.215 seconds with return value 0
Press any key to continue . . .
```

ENG IN 08:34:53 13-11-2024

11.Write a program for to copy one string to another.

```
#include <stdio.h>
#include <string.h>
int main() {
char source[100], destination[100];
printf("Enter the source string: ");
fgets(source, sizeof(source), stdin);
source[strcspn(source, "\n")] = '\0';
strcpy(destination, source);
printf("Destination string: %s\n", destination);
return 0;
}
```

Output:



C:\Users\cheru\OneDrive\Desktop\DAA programs\copy one string to another.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
selection sort.cpp bubble sort.cpp multiply two matrix.cpp palindrome or not.cpp copy one string to another.cpp
main () : int
C:\Users\cheru\OneDrive\Des
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 4 Col: 37 Sel: 0 Lines: 11 Length: 314 Insert Done parsing in 0.015 seconds
Search
ENG IN
08:37:29
13-11-2024

```
#include <stdio.h>
#include <string.h>
int main() {
char source[100], destination[100];
printf("Enter the source string: ");
fgets(source, sizeof(source), stdin);
source[strcspn(source, "\n")] = '\0';
strcpy(destination, source);
printf("Destination string: %s\n", destinat
return 0;
}
```

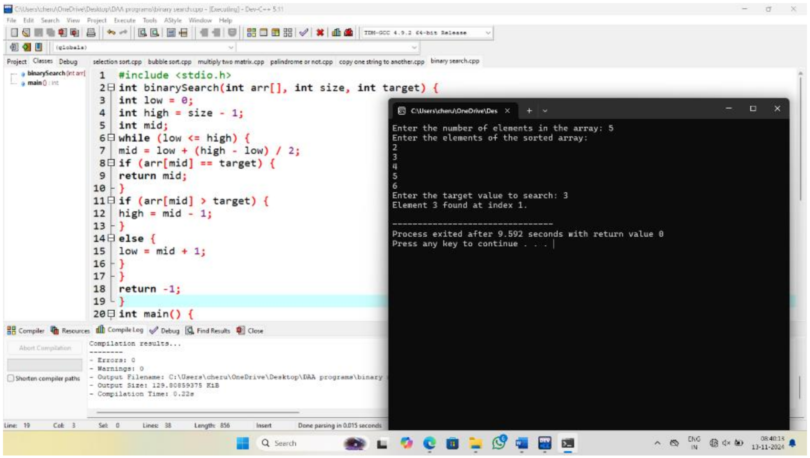
```
Enter the source string: saveetha
Destination string: saveetha

--------------------------------
Process exited after 6.205 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\copy on
- Output Size: 128.974609375 KiB
- Compilation Time: 0.24s
```

12.Write a Program to perform binary search.

```
#include <stdio.h>
int binarySearch(int arr[], int size, int target) {
int low = 0;
int high = size - 1;
int mid;
while (low <= high) {
mid = low + (high - low) / 2;
if (arr[mid] == target) {
return mid;
}
if (arr[mid] > target) {
high = mid - 1;
}
else {
low = mid + 1;
}
}
return -1;
}
int main() {
int n, target, result;
printf("Enter the number of elements in the array: ");
scanf("%d", &n);
int arr[n];
printf("Enter the elements of the sorted array:\n");
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
printf("Enter the target value to search: ");
scanf("%d", &target);
result = binarySearch(arr, n, target);
if (result != -1) {
printf("Element %d found at index %d.\n", target, result);
} else {
printf("Element %d not found in the array.\n", target);
}
return 0;
}
```

Output:



```
#include <stdio.h>
int binarySearch(int arr[], int size, int target) {
int low = 0;
int high = size - 1;
int mid;
while (low <= high) {
mid = low + (high - low) / 2;
if (arr[mid] == target) {
return mid;
}
if (arr[mid] > target) {
high = mid - 1;
}
else {
low = mid + 1;
}
}
return -1;
}
int main() {
```

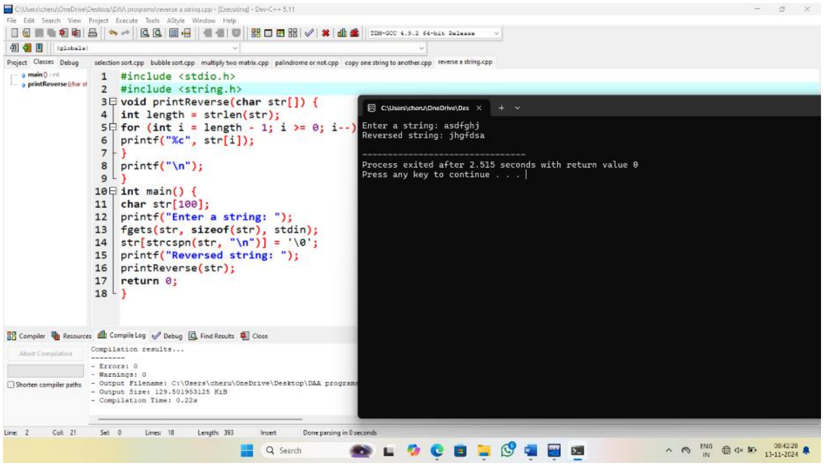
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA programs\binary
- Output Size: 129.80859375 KiB
- Compilation Time: 0.22s
```

```
Enter the number of elements in the array: 5
Enter the elements of the sorted array:
2
3
4
5
6
Enter the target value to search: 3
Element 3 found at index 1.

--------------------------------
Process exited after 9.592 seconds with return value 0
Press any key to continue . . .
```

13.Write a program to print the reverse of a string.

```
#include <stdio.h>
#include <string.h>
void printReverse(char str[]) {
int length = strlen(str);
for (int i = length - 1; i >= 0; i--) {
printf("%c", str[i]);
}
printf("\n");
}
int main() {
char str[100];
printf("Enter a string: ");
fgets(str, sizeof(str), stdin);
str[strcspn(str, "\n")] = '\0';
printf("Reversed string: ");
printReverse(str);
return 0;
}
```

Output:



```
#include <stdio.h>
#include <string.h>
void printReverse(char str[]) {
int length = strlen(str);
for (int i = length - 1; i >= 0; i--)
printf("%c", str[i]);
}
printf("\n");
}
int main() {
char str[100];
printf("Enter a string: ");
fgets(str, sizeof(str), stdin);
str[strcspn(str, "\n")] = '\0';
printf("Reversed string: ");
printReverse(str);
return 0;
}
```

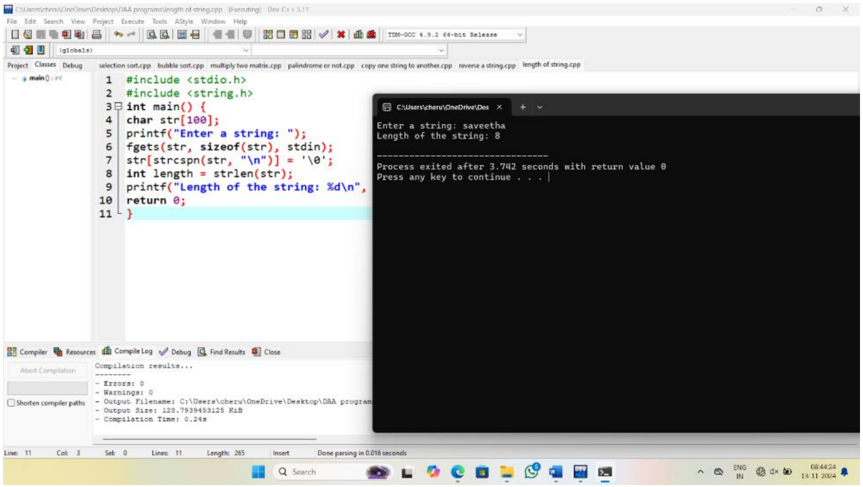
```
Enter a string: asdfghj
Reversed string: jhgfdsa

--------------------------------
Process exited after 2.515 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA program
- Output Size: 129.501953125 KiB
- Compilation Time: 0.22s
```

14.Write a program to find the length of a string.

```
#include <stdio.h>

#include <string.h>

int main() {

char str[100];

printf("Enter a string: ");

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

str[strcspn(str, "\n")] = '\0';

int length = strlen(str);

printf("Length of the string: %d\n", length);

return 0;

}
```

Output:



```
#include <stdio.h>
#include <string.h>
int main() {
char str[100];
printf("Enter a string: ");
fgets(str, sizeof(str), stdin);
str[strcspn(str, "\n")] = '\0';
int length = strlen(str);
printf("Length of the string: %d\n",
return 0;
}
```

```
Enter a string: saveetha
Length of the string: 8

--------------------------------
Process exited after 3.742 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\cheru\OneDrive\Desktop\DAA program
- Output Size: 128.7939453125 KiB
- Compilation Time: 0.24s
```

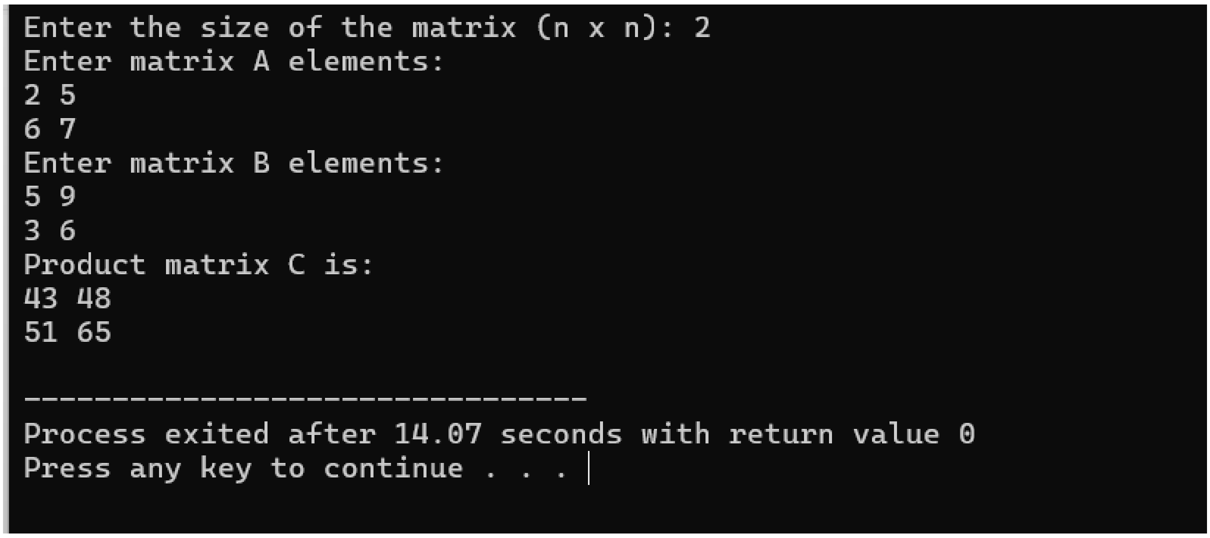
15.Write a program to perform Strassen’s Matrix Multiplication.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 2
void add(int A[SIZE][SIZE], int B[SIZE][SIZE], int result[SIZE][SIZE]) {
for (int i = 0; i < SIZE; i++) {
for (int j = 0; j < SIZE; j++) {
result[i][j] = A[i][j] + B[i][j];
}
}
}
void subtract(int A[SIZE][SIZE], int B[SIZE][SIZE], int
result[SIZE][SIZE])
{
for (int i = 0; i < SIZE; i++) {
for (int j = 0; j < SIZE; j++) {
result[i][j] = A[i][j] - B[i][j];
}
}
}
void strassen(int A[SIZE][SIZE], int B[SIZE][SIZE], int C[SIZE][SIZE]) {
if (SIZE == 1) {
C[0][0] = A[0][0] * B[0][0];
return;
}
// Allocate memory for submatrices
int A11[SIZE / 2][SIZE / 2], A12[SIZE / 2][SIZE / 2];
int A21[SIZE / 2][SIZE / 2], A22[SIZE / 2][SIZE / 2];
int B11[SIZE / 2][SIZE / 2], B12[SIZE / 2][SIZE / 2];
int B21[SIZE / 2][SIZE / 2], B22[SIZE / 2][SIZE / 2];
int M1[SIZE / 2][SIZE / 2], M2[SIZE / 2][SIZE / 2];
int M3[SIZE / 2][SIZE / 2], M4[SIZE / 2][SIZE / 2];
int M5[SIZE / 2][SIZE / 2], M6[SIZE / 2][SIZE / 2];
int M7[SIZE / 2][SIZE / 2];
int temp1[SIZE / 2][SIZE / 2], temp2[SIZE / 2][SIZE / 2];
for (int i = 0; i < SIZE / 2; i++) {
for (int j = 0; j < SIZE / 2; j++) {
A11[i][j] = A[i][j];
A12[i][j] = A[i][j + SIZE / 2];
A21[i][j] = A[i + SIZE / 2][j];
A22[i][j] = A[i + SIZE / 2][j + SIZE / 2];
      B11[i][j] = B[i][j];
      B12[i][j] = B[i][j + SIZE / 2];
      B21[i][j] = B[i + SIZE / 2][j];
      B22[i][j] = B[i + SIZE / 2][j + SIZE / 2];
    }
  }
  add(A11, A22, temp1);
  add(B11, B22, temp2);
  strassen(temp1, temp2, M1);

  add(A21, A22, temp1);
  strassen(temp1, B11, M2);
```

```
subtract(B12, B22, temp1);
strassen(A11, temp1, M3);

subtract(B21, B11, temp1);
strassen(A22, temp1, M4);

add(A11, A12, temp1);
strassen(temp1, B22, M5);

subtract(A21, A11, temp1);
add(B11, B12, temp2);
strassen(temp1, temp2, M6);

subtract(A12, A22, temp1);
add(B21, B22, temp2);
strassen(temp1, temp2, M7);
add(M1, M4, temp1);
subtract(temp1, M5, temp2);
add(temp2, M7, C);

add(M3, M5, temp1);
C[0][1] = temp1[0][0];

add(M2, M4, C);
C[1][0] = C[0][0];

add(M1, M3, temp1);
```

```
    subtract(temp1, M6, C);
    C[1][1] = C[0][0];
}

int main() {
    int A[SIZE][SIZE] = {{1, 2}, {3, 4}};
    int B[SIZE][SIZE] = {{5, 6}, {7, 8}};
    int C[SIZE][SIZE];
    strassen(A, B, C);
    printf("Result matrix C:\n");
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%d ", C[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:



```
Enter the size of the matrix (n x n): 2
Enter matrix A elements:
2 5
6 7
Enter matrix B elements:
5 9
3 6
Product matrix C is:
43 48
51 65

--------------------------------
Process exited after 14.07 seconds with return value 0
Press any key to continue . . .
```

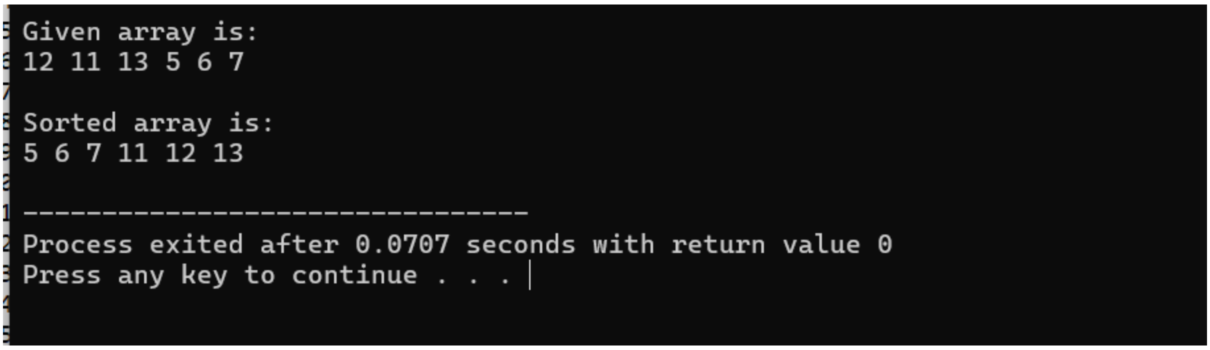
16.Write a program to perform Merge Sort.

```
#include <stdio.h>
void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int leftArr[n1], rightArr[n2];
    for (int i = 0; i< n1; i++) {
leftArr[i] = arr[left + i];
    }
    for (int i = 0; i< n2; i++) {
rightArr[i] = arr[mid + 1 + i];
    }
    int i = 0, j = 0, k = left;
    while (i< n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
arr[k] = leftArr[i];
i++;
        } else {
arr[k] = rightArr[j];
j++;
        }
        k++;
    }
    while (i< n1) {
arr[k] = leftArr[i];
i++;
        k++;
```

```
   }
   while (j < n2) {
arr[k] = rightArr[j];
j++;
      k++;
   }
}
void mergeSort(int arr[], int left, int right) {
   if (left < right) {
      int mid = left + (right - left) / 2;
mergeSort(arr, left, mid);
mergeSort(arr, mid + 1, right);
merge(arr, left, mid, right);
   }
}
void printArray(int arr[], int size) {
   for (int i = 0; i< size; i++) {
printf("%d ", arr[i]);
   }
printf("\n");
}
int main() {
   int arr[] = {12, 11, 13, 5, 6, 7};
   int arr_size = sizeof(arr) / sizeof(arr[0]);
printf("Given array is: \n");
printArray(arr, arr_size);
mergeSort(arr, 0, arr_size - 1);
```

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

Output:

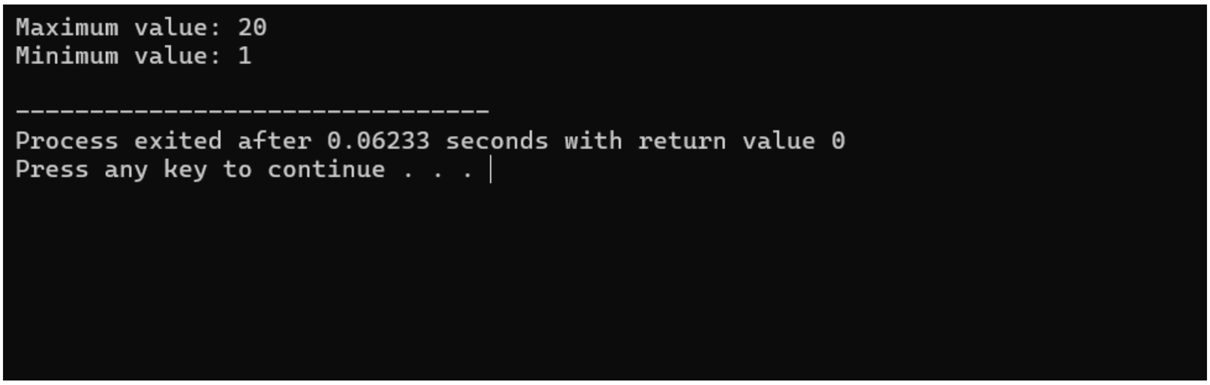


```
Given array is:
12 11 13 5 6 7

Sorted array is:
5 6 7 11 12 13

--------------------------------
Process exited after 0.0707 seconds with return value 0
Press any key to continue . . .
```

17.Using Divide and Conquer strategy to find Max and Min value in the list.

```
#include <stdio.h>
typedef struct {
    int max;
    int min;
} MaxMin;
MaxMinfindMaxMin(int arr[], int low, int high) {
MaxMin result, leftResult, rightResult;
    if (low == high) {
result.max = arr[low];
result.min = arr[low];
        return result;
    }
    int mid = (low + high) / 2;
leftResult = findMaxMin(arr, low, mid);
rightResult = findMaxMin(arr, mid + 1, high);
result.max = (leftResult.max>rightResult.max) ?leftResult.max :
rightResult.max;
result.min = (leftResult.min<rightResult.min) ?leftResult.min :
rightResult.min;
   return result;
}
int main() {
   int arr[] = {12, 5, 8, 20, 7, 15, 1};
   int n = sizeof(arr) / sizeof(arr[0]);
MaxMin result = findMaxMin(arr, 0, n - 1);
printf("Maximum value: %d\n", result.max);
printf("Minimum value: %d\n", result.min);
   return 0;
}
```

OUTPUT:



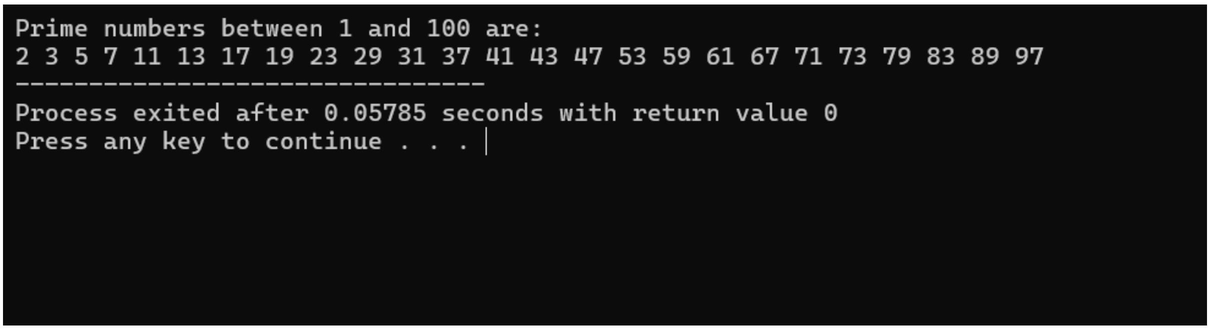
```
Maximum value: 20
Minimum value: 1

--------------------------------
Process exited after 0.06233 seconds with return value 0
Press any key to continue . . .
```

18.PRIME NUMBERS BETWEEN 1 AND 100

```
#include <stdio.h>
int isPrime(int num) {
   if (num<= 1) {
      return 0;
   }
```

```
    for (int i = 2; i * i<= num; i++) {
        if (num % i == 0) {
            return 0;
        }
    }
    return 1;
}
int main() {
printf("Prime numbers between 1 and 100 are:\n");

    for (int i = 1; i<= 100; i++) {
        if (isPrime(i)) {
printf("%d ", i);
        }
    }
    return 0;
}
```

OUTPUT:



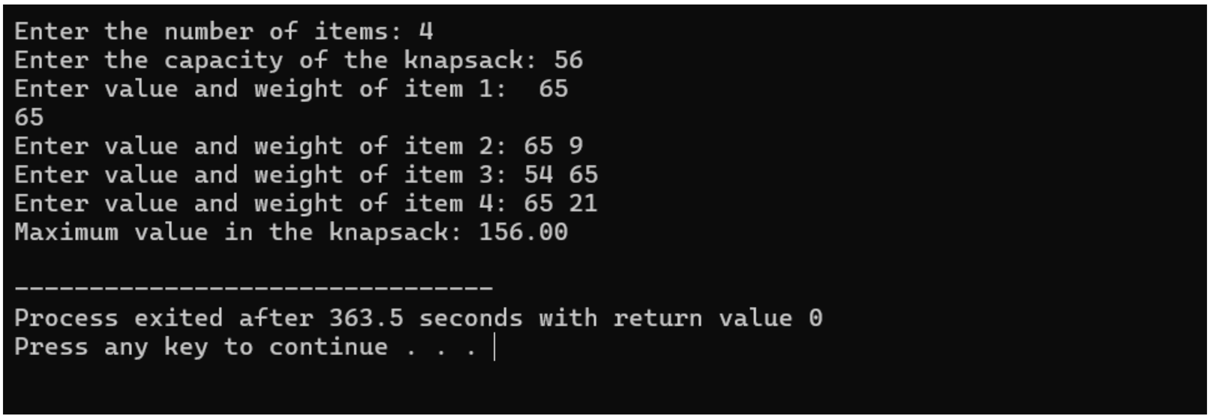
```
Prime numbers between 1 and 100 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
--------------------------------
Process exited after 0.05785 seconds with return value 0
Press any key to continue . . .
```

19.KNAPSACK PROBLEM USING GREEDY TECHNIQUES

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int weight;
    int value;
    float ratio;
} Item;
int compare(const void* a, const void* b) {
    Item* item1 = (Item*)a;
    Item* item2 = (Item*)b;
    return (item2->ratio > item1->ratio) - (item1->ratio > item2->ratio);
}
float fractionalKnapsack(int capacity, Item items[], int n) {
qsort(items, n, sizeof(Item), compare);
    int currentWeight = 0;
    float totalValue = 0.0;
    for (int i = 0; i< n; i++) {
        if (currentWeight + items[i].weight<= capacity) {
currentWeight += items[i].weight;
totalValue += items[i].value;
        } else {
            int remainingWeight = capacity - currentWeight;
totalValue += items[i].value * ((float)remainingWeight / items[i].weight);
            break;
        }
    }
    return totalValue;
}
int main() {
    int n, capacity;
```

```
printf("Enter the number of items: ");
scanf("%d", &n);
printf("Enter the capacity of the knapsack: ");
scanf("%d", &capacity);
   Item items[n];
   for (int i = 0; i< n; i++) {
printf("Enter value and weight of item %d: ", i + 1);
scanf("%d %d", &items[i].value, &items[i].weight);
      items[i].ratio = (float)items[i].value / items[i].weight;
   }
   float maxValue = fractionalKnapsack(capacity, items, n);
printf("Maximum value in the knapsack: %.2f\n", maxValue);
   return 0;
}
```

OUTPUT:



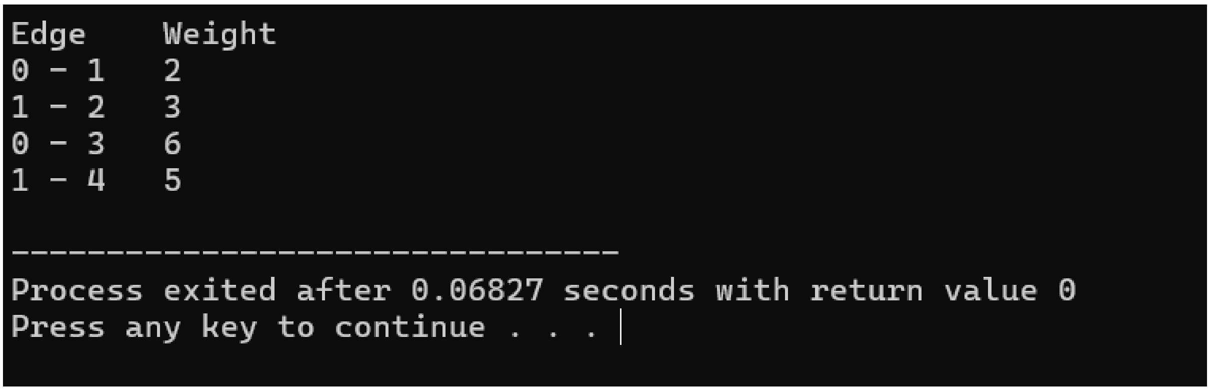
```
Enter the number of items: 4
Enter the capacity of the knapsack: 56
Enter value and weight of item 1:  65
65
Enter value and weight of item 2: 65 9
Enter value and weight of item 3: 54 65
Enter value and weight of item 4: 65 21
Maximum value in the knapsack: 156.00

--------------------------------
Process exited after 363.5 seconds with return value 0
Press any key to continue . . .
```

20.MST USING GREEDY TECHNIQUE

```
#include <stdio.h>
#include <limits.h>
#define V 5
int minKey(int key[], int mstSet[]) {
   int min = INT_MAX, min_index;
```

```
    for (int v = 0; v < V; v++)
        if (!mstSet[v] && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}
void primMST(int graph[V][V]) {
    int parent[V], key[V], mstSet[V] = {0};
    for (int i = 0; i< V; i++) key[i] = INT_MAX;
key[0] = 0, parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
mstSet[u] = 1;
        for (int v = 0; v < V; v++)
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
printf("Edge \tWeight\n");
    for (int i = 1; i< V; i++)
printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
}
int main() {
    int graph[V][V] = {
        {0, 2, 0, 6, 0},
        {2, 0, 3, 8, 5},
        {0, 3, 0, 0, 7},
        {6, 8, 0, 0, 9},
        {0, 5, 7, 9, 0}
    };
primMST(graph);
    return 0;
}
```

OUTPUT:

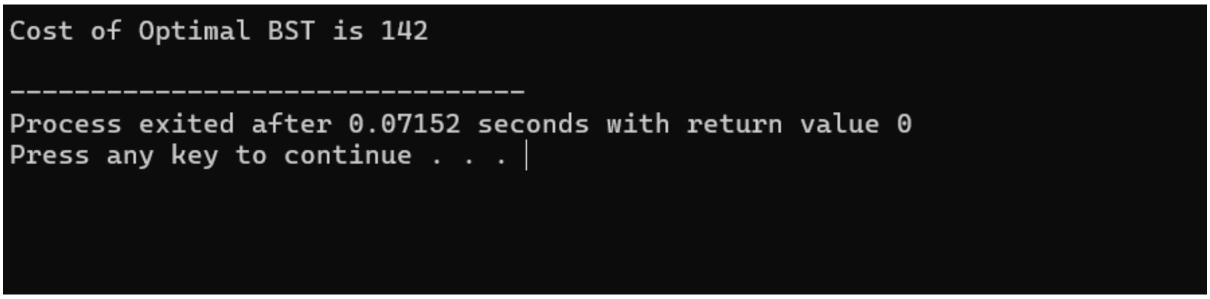


```
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5

--------------------------------
Process exited after 0.06827 seconds with return value 0
Press any key to continue . . .
```

21.OBST USING DYNAMIC PROGRAMMING

```
#include <stdio.h>
#include <limits.h>
int sum(int freq[], int i, int j) {
    int s = 0;
    for (int k = i; k <= j; k++)
        s += freq[k];
    return s;
}
int optimalBST(int keys[], int freq[], int n) {
    int cost[n][n];
    for (int i = 0; i< n; i++)
        cost[i][i] = freq[i];
    for (int len = 2; len<= n; len++) {
        for (int i = 0; i<= n - len; i++) {
            int j = i + len - 1;
            cost[i][j] = INT_MAX;
            int fsum = sum(freq, i, j);
            for (int r = i; r <= j; r++) {
                int c = ((r >i) ? cost[i][r - 1] : 0) +
                        ((r < j) ? cost[r + 1][j] : 0) + fsum;
                if (c < cost[i][j])
                    cost[i][j] = c;
            }
        }
    }
    return cost[0][n - 1];
}
int main() {
    int keys[] = {10, 12, 20};
    int freq[] = {34, 8, 50};
    int n = sizeof(keys) / sizeof(keys[0]);
printf("Cost of Optimal BST is %d\n", optimalBST(keys, freq, n));
    return 0;
}
```

OUTPUT:

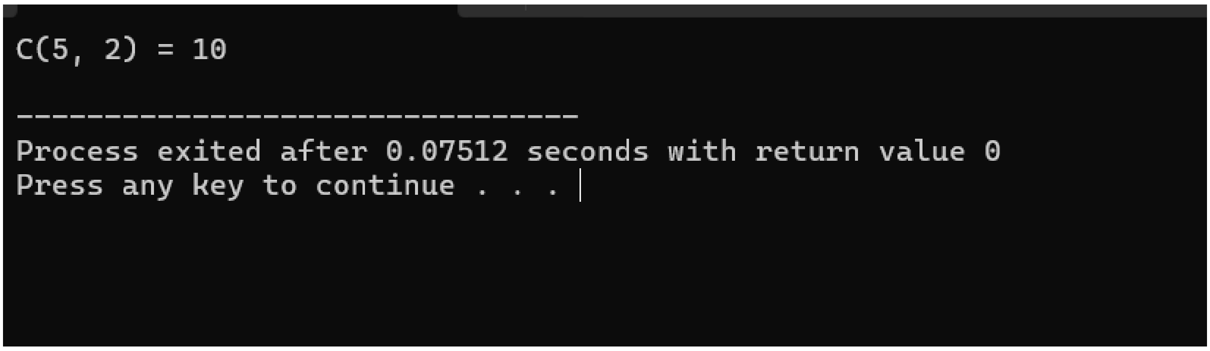


```
Cost of Optimal BST is 142

--------------------------------
Process exited after 0.07152 seconds with return value 0
Press any key to continue . . .
```

22.Using Dynamic programming techniques to find binomial coefficient of a given number

```
#include <stdio.h>

int binomialCoeff(int n, int k) {
    int C[n + 1][k + 1];
    for (int i = 0; i<= n; i++) {
        for (int j = 0; j <= (i<k ?i : k); j++) {
            if (j == 0 || j == i)
                C[i][j] = 1;
            else
                C[i][j] = C[i - 1][j - 1] + C[i - 1][j];
        }
    }
    return C[n][k];
}
int main() {
    int n = 5, k = 2;
printf("C(%d, %d) = %d\n", n, k, binomialCoeff(n, k));
    return 0;
}
```

OUTPUT:

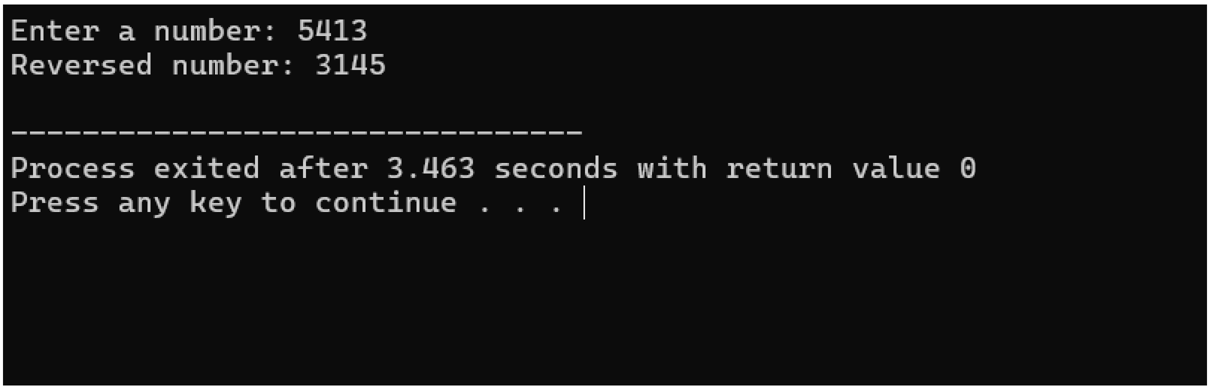


```
C(5, 2) = 10

--------------------------------
Process exited after 0.07512 seconds with return value 0
Press any key to continue . . .
```

23.Write a program to find the reverse of a given number.

```
#include <stdio.h>
int main() {
    int num, reversed = 0;
printf("Enter a number: ");
scanf("%d", &num);

    while (num != 0) {
        reversed = reversed * 10 + num % 10;
num /= 10;
    }
printf("Reversed number: %d\n", reversed);
    return 0;
}
```

OUTPUT:

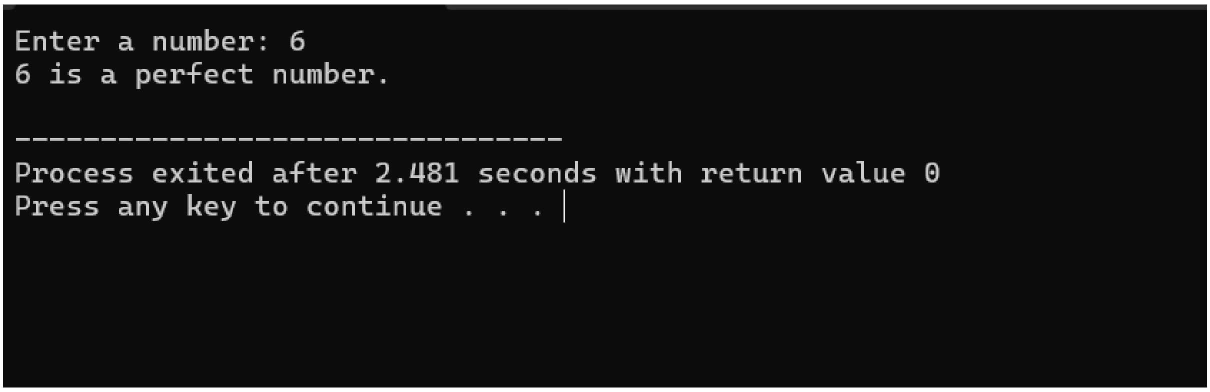


```
Enter a number: 5413
Reversed number: 3145

--------------------------------
Process exited after 3.463 seconds with return value 0
Press any key to continue . . .
```

24.Write a program to find the perfect number.

```
#include <stdio.h>
int main() {
    int num, sum = 0;
printf("Enter a number: ");
scanf("%d", &num);

   for (int i = 1; i<num; i++) {
      if (num % i == 0)
         sum += i;
   }
   if (sum == num)
printf("%d is a perfect number.\n", num);
   else
printf("%d is not a perfect number.\n", num);
   return 0;
}
```

OUTPUT:



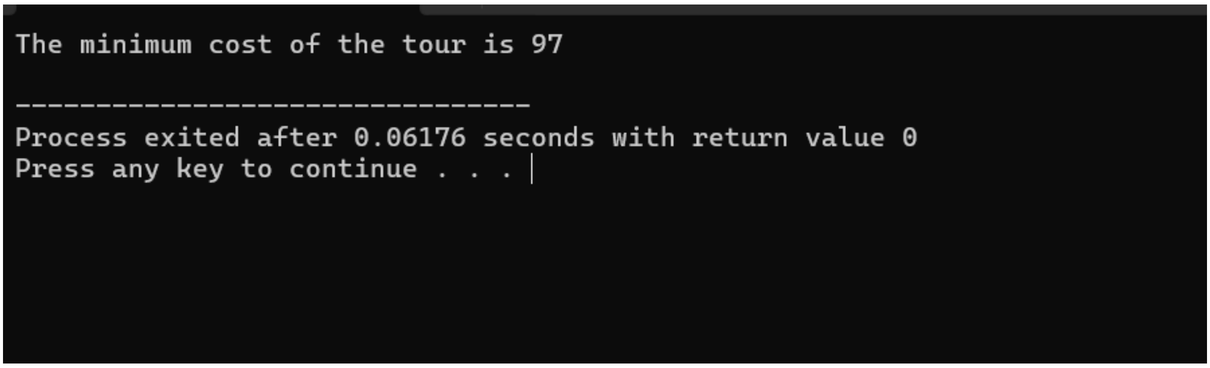
```
Enter a number: 6
6 is a perfect number.

--------------------------------
Process exited after 2.481 seconds with return value 0
Press any key to continue . . .
```

25.Write a program to perform travelling salesman problem using dynamic programming.

```
#include <stdio.h>
#include <limits.h>
#define N 4
#define INF INT_MAX
```

```
int dist[N][N] = {
    {0, 20, 42, 35},
    {20, 0, 30, 34},
    {42, 30, 0, 12},
    {35, 34, 12, 0}
};
int dp[1 << N][N];
int tsp(int mask, int pos) {
    if (mask == ((1 << N) - 1))
        return dist[pos][0];
    if (dp[mask][pos] != -1)
        return dp[mask][pos];
    int ans = INF;
    for (int city = 0; city < N; city++) {
        if (!(mask & (1 << city))) {
            int newAns = dist[pos][city] + tsp(mask | (1 << city), city);
            if (newAns<ans)
ans = newAns;
        }
    }
    return dp[mask][pos] = ans;
}
int main() {
    for (int i = 0; i< (1 << N); i++)
        for (int j = 0; j < N; j++)
dp[i][j] = -1;
    int result = tsp(1, 0);
printf("The minimum cost of the tour is %d\n", result);
   return 0;
}
```

OUTPUT:



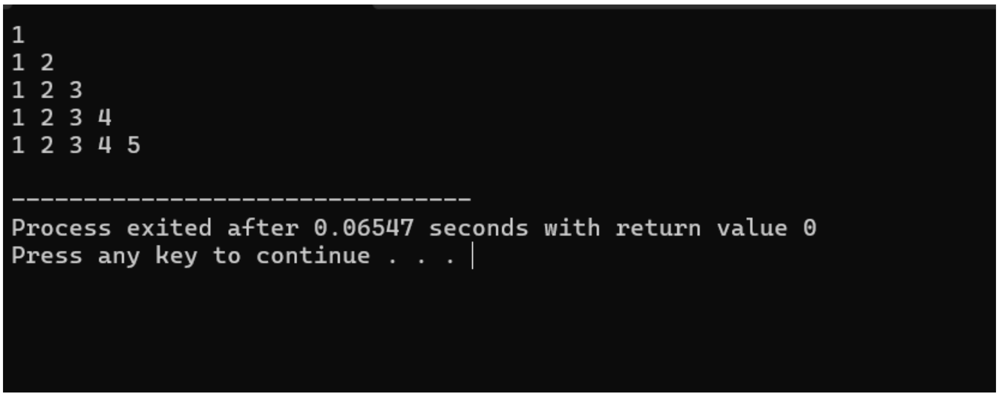
```
The minimum cost of the tour is 97

--------------------------------
Process exited after 0.06176 seconds with return value 0
Press any key to continue . . .
```

26.Write a program for the given pattern If n=4

```
     1
     1 2
     1 2 3
     1 2 3 4
```

```
#include <stdio.h>
int main() {
   for (int i = 1; i<= 5; i++)
      for (int j = 1; j <= i; j++)
printf("%d ", j);
printf("\n");
   return 0;
}
```

OUTPUT:



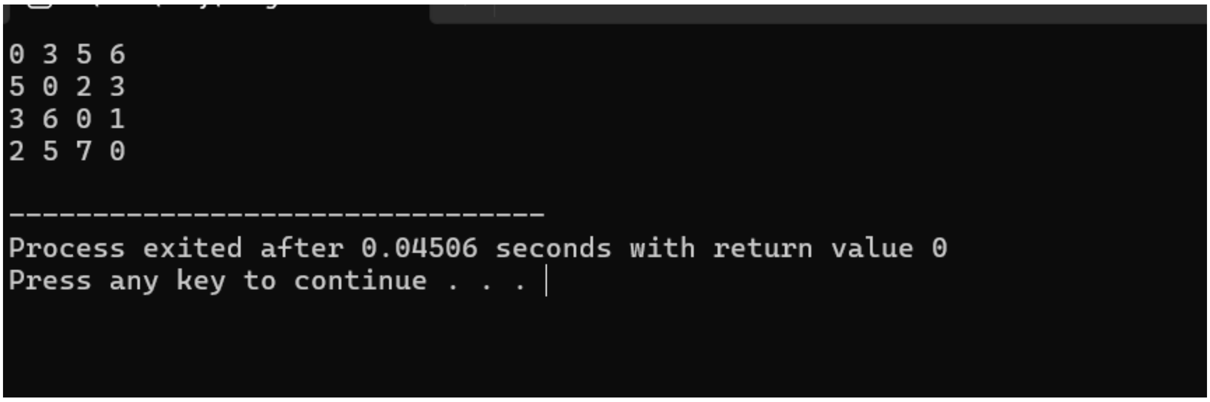
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

--------------------------------
Process exited after 0.06547 seconds with return value 0
Press any key to continue . . .
```

27.Write a program to perform Floyd's algorithm.

```
#include <stdio.h>
#define INF 99999
#define V 4
void floydWarshall(int graph[V][V]) {
  int dist[V][V], i, j, k;
  for (i = 0; i< V; i++) {
    for (j = 0; j < V; j++) {
dist[i][j] = graph[i][j];
    }
  }
  for (k = 0; k < V; k++) {
    for (i = 0; i< V; i++) {
      for (j = 0; j < V; j++) {
        if (dist[i][k] + dist[k][j] <dist[i][j]) {
dist[i][j] = dist[i][k] + dist[k][j];
        }
      }
    }
  }
```

```
    for (i = 0; i< V; i++) {
        for (j = 0; j < V; j++) {
            if (dist[i][j] == INF) printf("INF ");
            else printf("%d ", dist[i][j]);
        }
printf("\n");
    }
}
int main() {
    int graph[V][V] = {
        {0, 3, INF, 7},
        {8, 0, 2, INF},
        {5, INF, 0, 1},
        {2, INF, INF, 0}
    };
floydWarshall(graph);
    return 0;
}
```

OUTPUT:



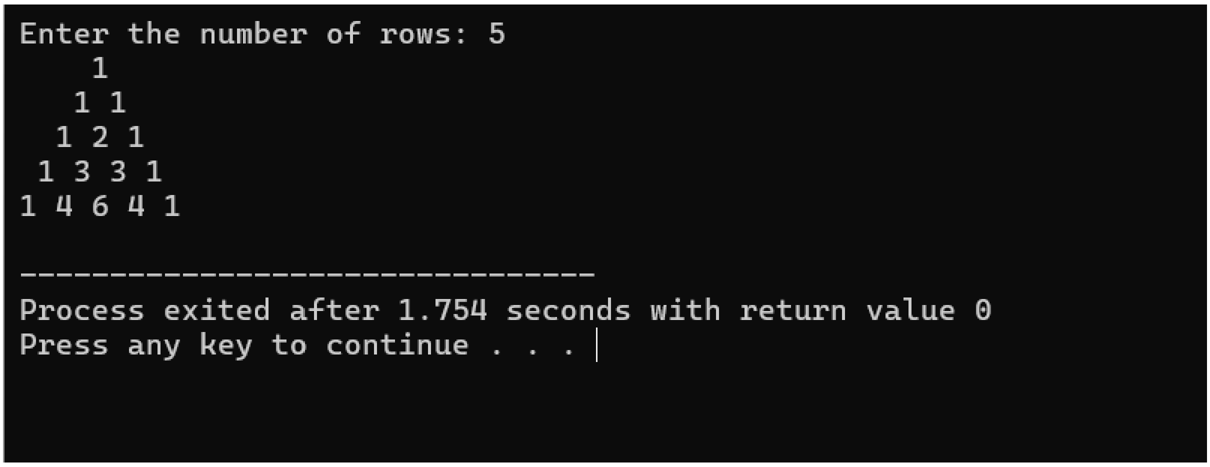
```
0 3 5 6
5 0 2 3
3 6 0 1
2 5 7 0

--------------------------------
Process exited after 0.04506 seconds with return value 0
Press any key to continue . . .
```

28.Write a program for pascal triangle.

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

```
int main() {
    int n, i, j, num;
printf("Enter the number of rows: ");
scanf("%d", &n);
    for (i = 0; i< n; i++) {
num = 1;
        for (j = 0; j < n - i - 1; j++) {
printf(" ");
        }
        for (j = 0; j <= i; j++) {
printf("%d ", num);
num = num * (i - j) / (j + 1);
        }
printf("\n");
    }
    return 0;
}
```

OUTPUT:



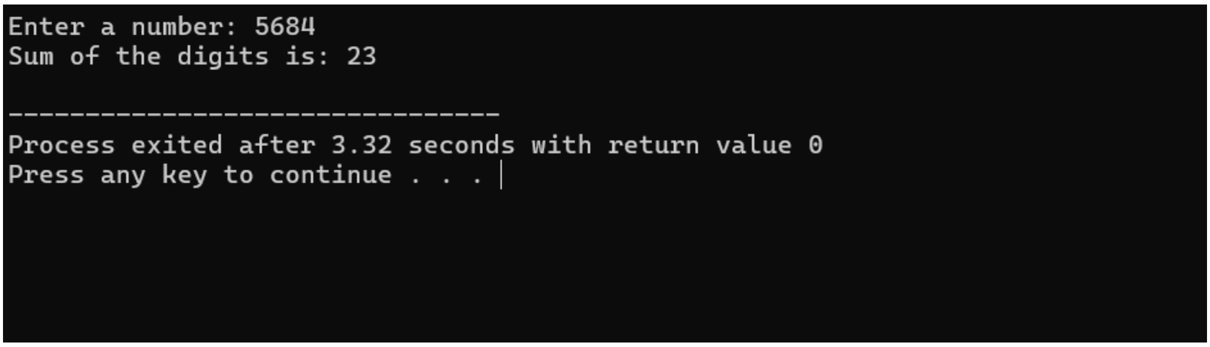
```
Enter the number of rows: 5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1

--------------------------------
Process exited after 1.754 seconds with return value 0
Press any key to continue . . .
```

29.SUM OF DIDGITS

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

```
int main() {
   int num, sum = 0, digit;
printf("Enter a number: ");
scanf("%d", &num);
   while (num != 0) {
      digit = num % 10;
      sum += digit;
num = num / 10;
   }
printf("Sum of the digits is: %d\n", sum);
   return 0;
}
```

OUTPUT:

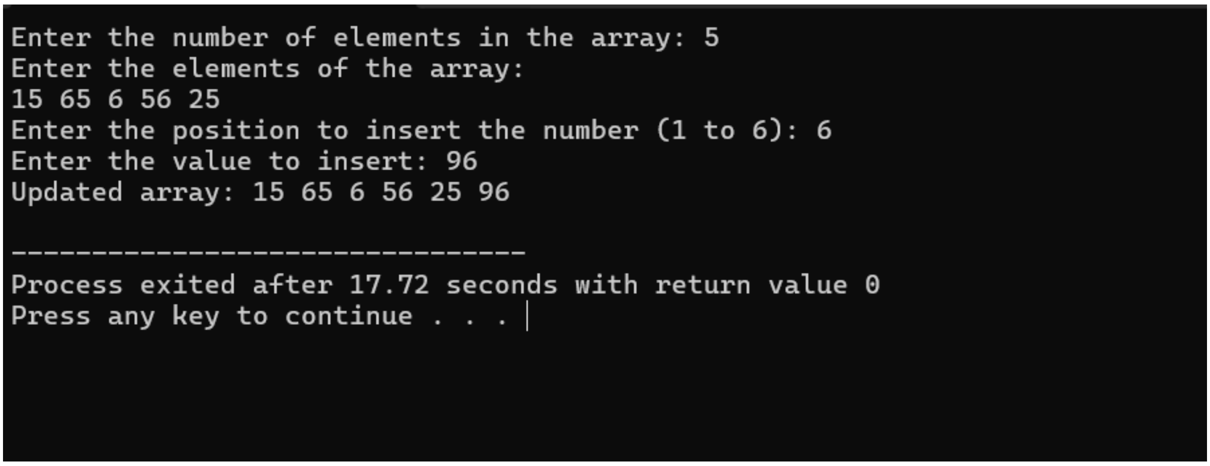


```
Enter a number: 5684
Sum of the digits is: 23

--------------------------------
Process exited after 3.32 seconds with return value 0
Press any key to continue . . .
```

30.INSERT A NUMBER IN THE LIST

```
#include <stdio.h>
int main() {
   int arr[100], n, i, position, value;
printf("Enter the number of elements in the array: ");
scanf("%d", &n);
printf("Enter the elements of the array: \n");
   for (i = 0; i< n; i++) {
scanf("%d", &arr[i]);
   }
printf("Enter the position to insert the number (1 to %d): ", n + 1);
scanf("%d", &position);
printf("Enter the value to insert: ");
scanf("%d", &value);
   for (i = n; i>= position; i--) {
arr[i] = arr[i - 1];
   }
arr[position - 1] = value;
   n++;
printf("Updated array: ");
   for (i = 0; i< n; i++) {
printf("%d ", arr[i]);
   }
printf("\n");
   return 0;
}
```

OUTPUT:

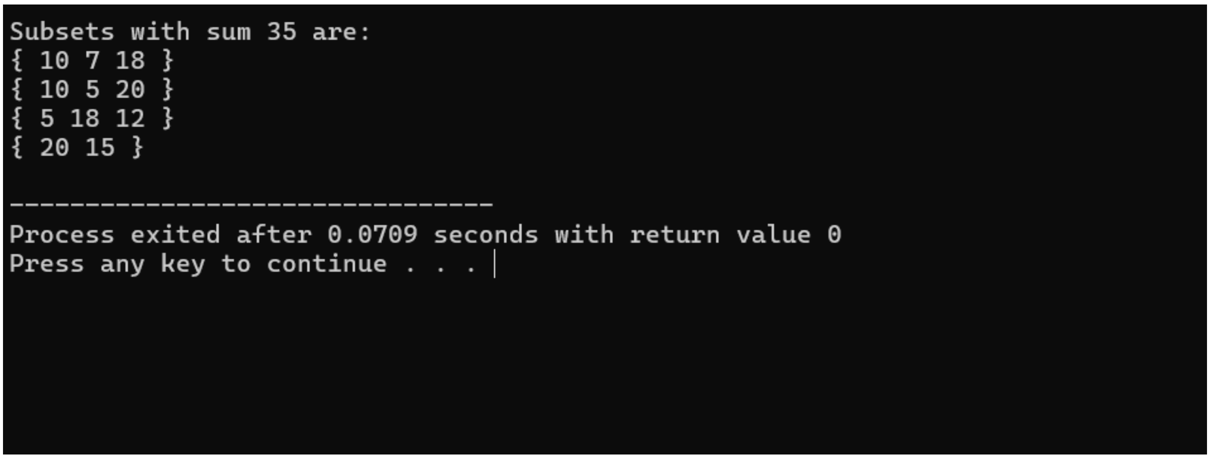


```
Enter the number of elements in the array: 5
Enter the elements of the array:
15 65 6 56 25
Enter the position to insert the number (1 to 6): 6
Enter the value to insert: 96
Updated array: 15 65 6 56 25 96

--------------------------------
Process exited after 17.72 seconds with return value 0
Press any key to continue . . .
```

31.SUM OF SUBSETS USING BACKTRACKING

```
#include <stdio.h>
void subsetSum(int arr[], int n, int target_sum, int index, int current_sum,
int current_subset[], int subset_size) {
   if (current_sum == target_sum) {
printf("{ ");
      for (int i = 0; i<subset_size; i++) {
printf("%d ", current_subset[i]);
      }
printf("}\n");
      return;
   }
   if (current_sum>target_sum || index == n) {
      return;
   }
current_subset[subset_size] = arr[index];
subsetSum(arr, n, target_sum, index + 1, current_sum + arr[index],
current_subset, subset_size + 1);
subsetSum(arr, n, target_sum, index + 1, current_sum, current_subset,
subset_size);
}
void findAllSubsets(int arr[], int n, int target_sum) {
   int current_subset[n];
subsetSum(arr, n, target_sum, 0, 0, current_subset, 0);
}
int main() {
   int arr[] = {10, 7, 5, 18, 12, 20, 15};
   int target_sum = 35;
   int n = sizeof(arr) / sizeof(arr[0]);
printf("Subsets with sum %d are:\n", target_sum);
findAllSubsets(arr, n, target_sum);
   return 0;
}
```

OUTPUT:



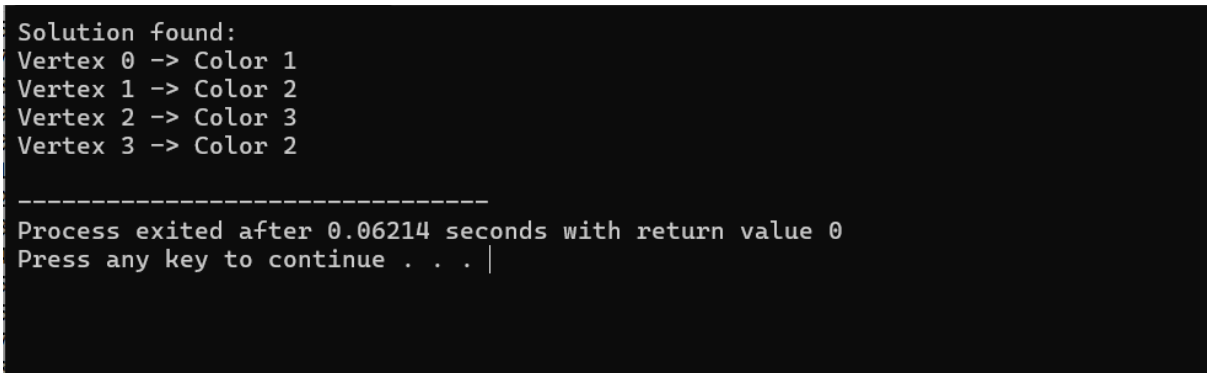
```
Subsets with sum 35 are:
{ 10 7 18 }
{ 10 5 20 }
{ 5 18 12 }
{ 20 15 }

--------------------------------
Process exited after 0.0709 seconds with return value 0
Press any key to continue . . .
```

32.GRAPH COLOURING USING BACKTRACKING

```
#include <stdio.h>
#include <stdbool.h>
#define N 4
bool isSafe(int vertex, int graph[N][N], int colors[], int color) {
    for (int i = 0; i< N; i++) {
        if (graph[vertex][i] &&colors[i] == color) {
            return false;
        }
    }
    return true;
}
bool graphColoring(int graph[N][N], int m, int colors[], int vertex) {
    if (vertex == N) {
        return true;
    }
    for (int color = 1; color<= m; color++) {
        if (isSafe(vertex, graph, colors, color)) {
colors[vertex] = color;
            if (graphColoring(graph, m, colors, vertex + 1)) {
                return true;
            }
colors[vertex] = 0;
        }
    }
    return false;
}
void solveGraphColoring(int graph[N][N], int m) {
    int colors[N] = {0};
    if (graphColoring(graph, m, colors, 0)) {
printf("Solution found:\n");
        for (int i = 0; i< N; i++) {
```

```
printf("Vertex %d ->Color %d\n", i, colors[i]);
        }
    } else {
printf("No solution exists\n");
    }
}
int main() {
    int graph[N][N] = {
        {0, 1, 1, 1},
        {1, 0, 1, 0},
        {1, 1, 0, 1},
        {1, 0, 1, 0}
    };
    int m = 3;
solveGraphColoring(graph, m);
    return 0;
}
```

OUTPUT:



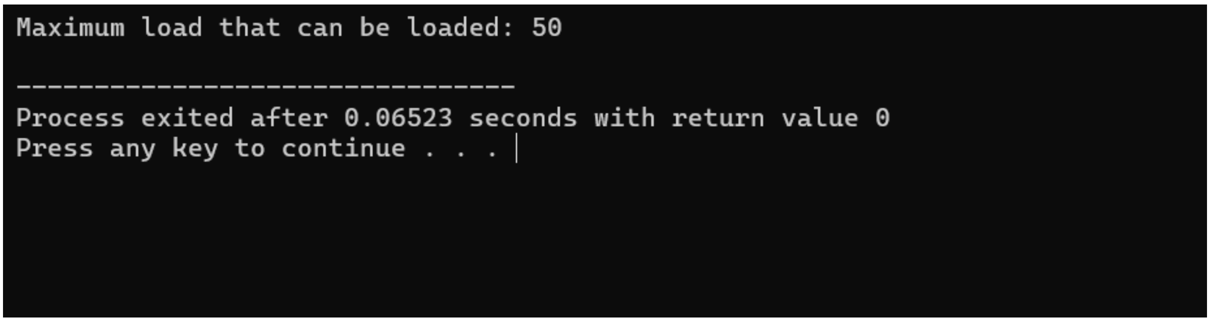
```
Solution found:
Vertex 0 -> Color 1
Vertex 1 -> Color 2
Vertex 2 -> Color 3
Vertex 3 -> Color 2

--------------------------------
Process exited after 0.06214 seconds with return value 0
Press any key to continue . . .
```

33.CONTAINER LOADING PROBLEM

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

```
int maxLoad = 0;
void backtrack(int weights[], int n, int capacity, int index, int currentLoad)
{
    if (currentLoad> capacity) {
        return;
    }
    if (currentLoad>maxLoad) {
maxLoad = currentLoad;
    }
    if (index == n) {
        return;
    }
backtrack(weights, n, capacity, index + 1, currentLoad + weights[index]);
backtrack(weights, n, capacity, index + 1, currentLoad);
}
int maxContainerLoad(int weights[], int n, int capacity) {
maxLoad = 0;
backtrack(weights, n, capacity, 0, 0);
    return maxLoad;
}
int main() {
    int weights[] = {10, 20, 30, 40};
    int n = sizeof(weights) / sizeof(weights[0]);
    int capacity = 50;
    int maxLoadPossible = maxContainerLoad(weights, n, capacity);
printf("Maximum load that can be loaded: %d\n", maxLoadPossible);
    return 0;
}
```

OUTPUT:

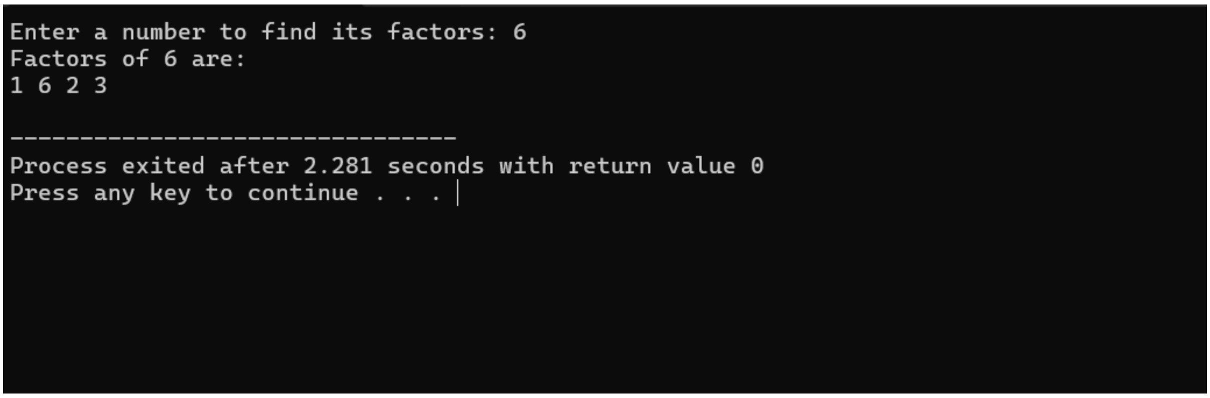


```
Maximum load that can be loaded: 50

--------------------------------
Process exited after 0.06523 seconds with return value 0
Press any key to continue . . .
```

34.LIST OF ALL FACTORS FOR N VALUE

```
#include <stdio.h>
#include <math.h>
void findFactors(int n) {
printf("Factors of %d are:\n", n);
   for (int i = 1; i<= sqrt(n); i++) {
      if (n % i == 0) {
printf("%d ", i);
         if (i != n / i) {
printf("%d ", n / i);
         }
      }
   }
printf("\n");
}
int main() {
   int n;
printf("Enter a number to find its factors: ");
scanf("%d", &n);
findFactors(n);
   return 0;
}
```

OUTPUT:



```
Enter a number to find its factors: 6
Factors of 6 are:
1 6 2 3

--------------------------------
Process exited after 2.281 seconds with return value 0
Press any key to continue . . .
```

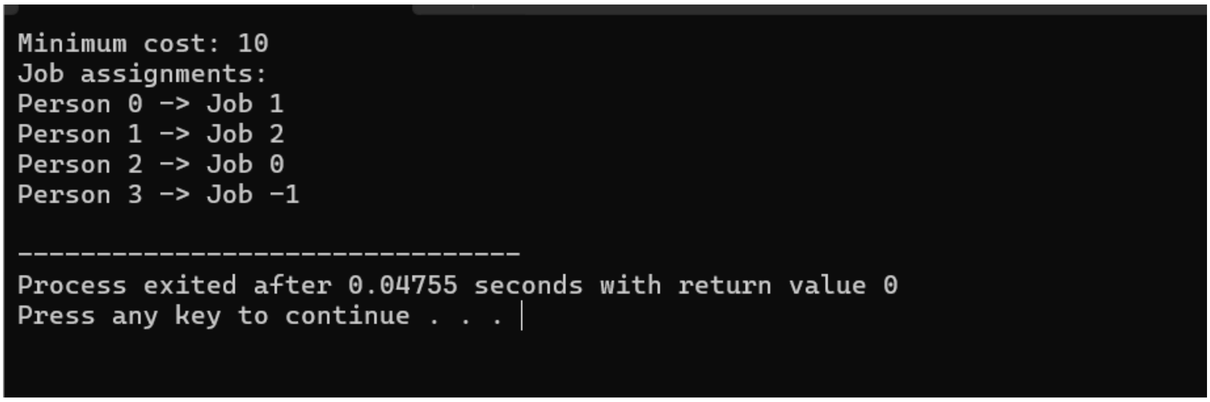
35.JOB ASSIGNMENT PROBLEM USING BRANCH AND BOUND

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#define N 4
typedef struct Node {
   int cost;
   int lowerBound;
   int jobAssignment[N];
   bool assigned[N];
   int level;
} Node;
int calculateLowerBound(int costMatrix[N][N], bool assigned[N], int level)
{
   int lowerBound = 0;
   for (int i = level; i< N; i++) {
      int minCost = INT_MAX;
      for (int j = 0; j < N; j++) {
         if (!assigned[j] &&costMatrix[i][j] <minCost) {
minCost = costMatrix[i][j];
         }
      }
lowerBound += minCost;
   }
   return lowerBound;
}
void branchAndBound(int costMatrix[N][N]) {
   int minCost = INT_MAX;
   Node bestNode;
   Node root;
root.cost = 0;
root.level = 0;
   for (int i = 0; i< N; i++) {
root.assigned[i] = false;
root.jobAssignment[i] = -1;
   }
root.lowerBound = calculateLowerBound(costMatrix, root.assigned,
root.level);
   Node queue[N * N];
   int queueSize = 0;
   queue[queueSize++] = root;
   while (queueSize> 0) {
      Node currentNode = queue[--queueSize];
```

```
    if (currentNode.lowerBound>= minCost) continue;
    if (currentNode.level == N) {
      if (currentNode.cost<minCost) {
minCost = currentNode.cost;
bestNode = currentNode;
      }
      continue;
    }
    for (int job = 0; job < N; job++) {
      if (!currentNode.assigned[job]) {
        Node newNode = currentNode;
newNode.level++;
newNode.jobAssignment[currentNode.level - 1] = job;
newNode.cost += costMatrix[currentNode.level - 1][job];
newNode.assigned[job] = true;
newNode.lowerBound = newNode.cost + calculateLowerBound(costMatrix,
newNode.assigned, newNode.level);
        if (newNode.lowerBound<minCost) {
          queue[queueSize++] = newNode;
        }
      }
    }
  }
printf("Minimum cost: %d\n", minCost);
printf("Job assignments:\n");
  for (int i = 0; i< N; i++) {
printf("Person %d -> Job %d\n", i, bestNode.jobAssignment[i]);
  }
```

```
}
int main() {
   int costMatrix[N][N] = {
      {9, 2, 7, 8},
      {6, 4, 3, 7},
      {5, 8, 1, 8},
      {7, 6, 9, 4}
   };
branchAndBound(costMatrix);
   return 0;
}
```

OUTPUT:

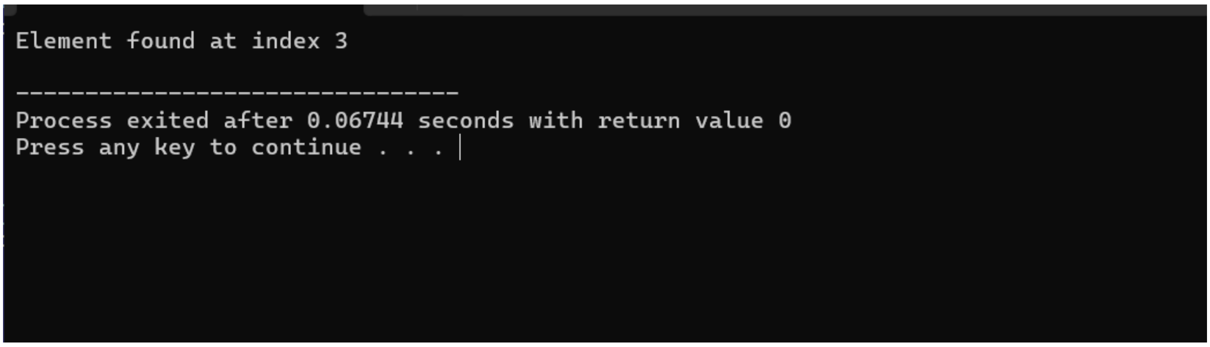


```
Minimum cost: 10
Job assignments:
Person 0 -> Job 1
Person 1 -> Job 2
Person 2 -> Job 0
Person 3 -> Job -1

--------------------------------
Process exited after 0.04755 seconds with return value 0
Press any key to continue . . .
```

36.LINEAR SEARCH

```
#include <stdio.h>
int linearSearch(int arr[], int n, int target) {
   for (int i = 0; i< n; i++) {
      if (arr[i] == target) {
           return i;
      }
   }
   return -1;
}
int main() {
   int arr[] = {34, 21, 56, 78, 90, 23, 12};
   int n = sizeof(arr) / sizeof(arr[0]);
   int target = 78;
   int result = linearSearch(arr, n, target);
   if (result != -1) {
printf("Element found at index %d\n", result);
   } else {
printf("Element not found in the array\n");
   }
   return 0;
}
```

OUTPUT:



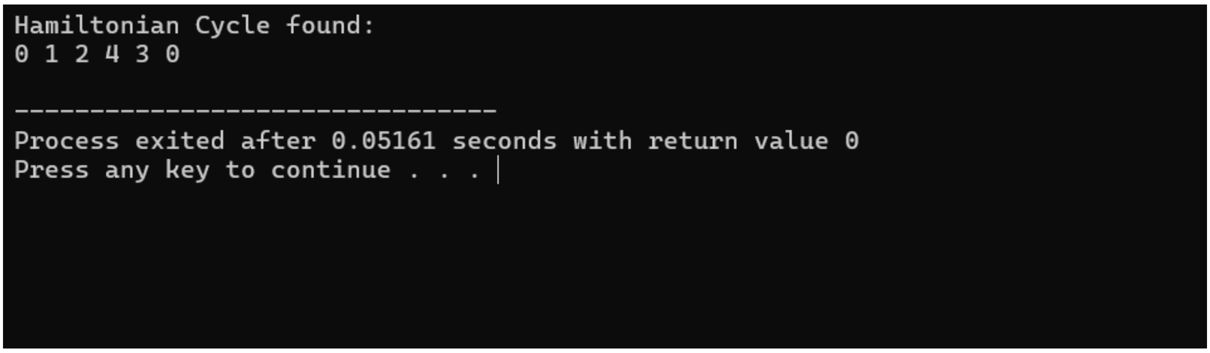
```
Element found at index 3

--------------------------------
Process exited after 0.06744 seconds with return value 0
Press any key to continue . . .
```

37.HAMILTONIAN CIRCUIT USING BACKTRACKING

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

```
#include <stdbool.h>
#define V 5
bool canAddToPath(int v, int graph[V][V], int path[], int position) {
    if (graph[path[position - 1]][v] == 0)
        return false;
    for (int i = 0; i< position; i++) {
        if (path[i] == v)
            return false;
    }
    return true;
}
bool hamiltonianCycle(int graph[V][V], int path[], int position) {
    if (position == V) {
        if (graph[path[position - 1]][path[0]] == 1)
            return true;
        else
            return false;
    }
    for (int v = 1; v < V; v++) {
        if (canAddToPath(v, graph, path, position)) {
            path[position] = v;
            if (hamiltonianCycle(graph, path, position + 1))
                return true;
            path[position] = -1;
        }
    }
    return false;
}
int main() {
    int graph[V][V] = {
        {0, 1, 0, 1, 0},
        {1, 0, 1, 1, 0},
        {0, 1, 0, 1, 1},
        {1, 1, 1, 0, 1},
        {0, 0, 1, 1, 0}
    };
    int path[V];
    for (int i = 0; i< V; i++) {
        path[i] = -1;
    }
path[0] = 0;
    if (hamiltonianCycle(graph, path, 1)) {
printf("Hamiltonian Cycle found: \n");
        for (int i = 0; i< V; i++) {
printf("%d ", path[i]);
        }
printf("%d\n", path[0]);
    } else {
printf("No Hamiltonian Cycle found\n");
    }
    return 0;
}
```

OUTPUT:



```
Hamiltonian Cycle found:
0 1 2 4 3 0

--------------------------------
Process exited after 0.05161 seconds with return value 0
Press any key to continue . . .
```

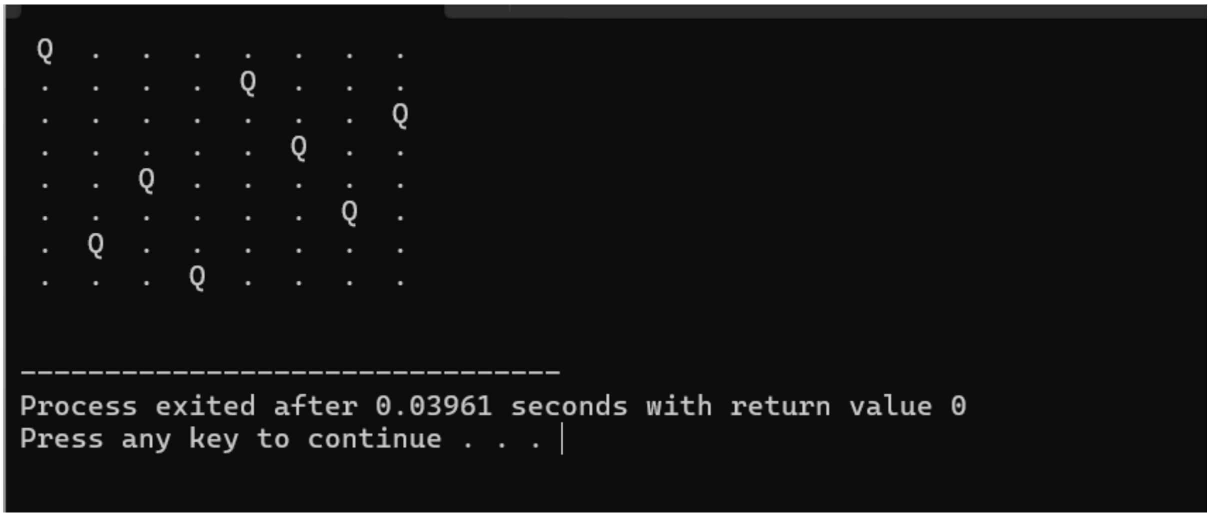
38.N QUEENS PROBLEM

```
#include <stdio.h>
#include <stdbool.h>
#define N 8
int board[N][N];
void printSolution() {
   for (int i = 0; i< N; i++) {
      for (int j = 0; j < N; j++) {
         if (board[i][j] == 1)
printf(" Q ");
         else
printf(" . ");
      }
printf("\n");
   }
printf("\n");
}
bool isSafe(int row, int col) {
   for (int i = 0; i< row; i++) {
      if (board[i][col] == 1)
         return false;
```

```
    }
    for (int i = row, j = col; i>= 0 && j >= 0; i--, j--) {
        if (board[i][j] == 1)
            return false;
    }
    for (int i = row, j = col; i>= 0 && j < N; i--, j++) {
        if (board[i][j] == 1)
            return false;
    }
    return true;
}
bool solveNQueens(int row) {
    if (row == N)
        return true;
    for (int col = 0; col < N; col++) {
        if (isSafe(row, col)) {
            board[row][col] = 1;
            if (solveNQueens(row + 1))
                return true;
            board[row][col] = 0;
        }
    }
    return false;
}
int main() {
    for (int i = 0; i< N; i++)
        for (int j = 0; j < N; j++)
```

```
          board[i][j] = 0;
    if (solveNQueens(0)) {
printSolution();
    } else {
printf("No solution exists\n");
    }
    return 0;
}
```

OUTPUT:

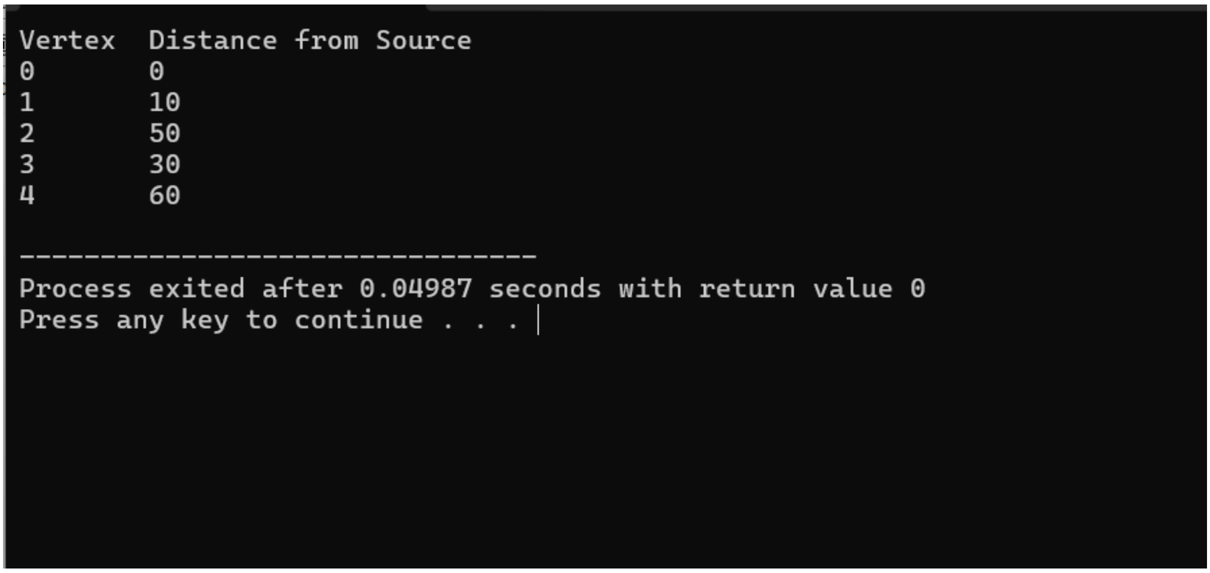


```
 Q  .  .  .  .  .  .  .
 .  .  .  .  Q  .  .  .
 .  .  .  .  .  .  .  Q
 .  .  .  .  .  Q  .  .
 .  .  Q  .  .  .  .  .
 .  .  .  .  .  .  Q  .
 .  Q  .  .  .  .  .  .
 .  .  .  Q  .  .  .  .


--------------------------------
Process exited after 0.03961 seconds with return value 0
Press any key to continue . . .
```

39.OPTIMAL COST BY USING APPROPRIATE ALGORITHM

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#define V 5
#define INF INT_MAX
void dijkstra(int graph[V][V], int src) {
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i< V; i++) {
dist[i] = INF;
sptSet[i] = false;
    }
dist[src] = 0;
    for (int count = 0; count < V - 1; count++) {
        int u = -1;
        for (int v = 0; v < V; v++) {
            if (!sptSet[v] && (u == -1 || dist[v] <dist[u])) {
                u = v;
            }
        }
sptSet[u] = true;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !sptSet[v] &&dist[u] != INF &&dist[u] +
graph[u][v] <dist[v]) {
dist[v] = dist[u] + graph[u][v];
            }
        }
    }
printf("Vertex\tDistance from Source\n");
    for (int i = 0; i< V; i++) {
printf("%d\t%d\n", i, dist[i]);
    }
}
int main() {
    int graph[V][V] = {
        {0, 10, 0, 30, 0},
        {10, 0, 50, 0, 0},
        {0, 50, 0, 20, 10},
        {30, 0, 20, 0, 60},
        {0, 0, 10, 60, 0}
    };

dijkstra(graph, 0);

    return 0;
}
```

OUTPUT:

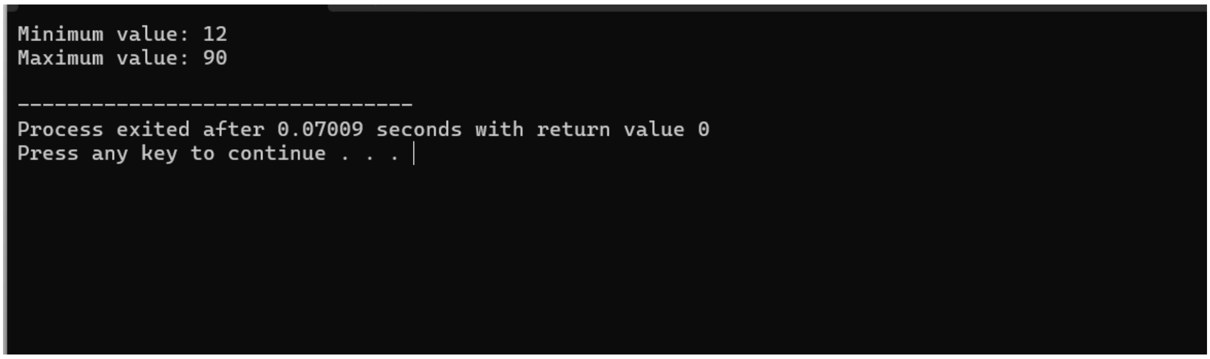


```
Vertex  Distance from Source
0       0
1       10
2       50
3       30
4       60

--------------------------------
Process exited after 0.04987 seconds with return value 0
Press any key to continue . . .
```

40.MIN MAX VALUE SEPERATELY FOR ALL NUMBERS IN THE LIST

```
#include <stdio.h>
void findMinMax(int numbers[], int size, int* min, int* max) {
    *min = numbers[0];
    *max = numbers[0];
    for (int i = 1; i< size; i++) {
        if (numbers[i] < *min) {
            *min = numbers[i];
        }
        if (numbers[i] > *max) {
            *max = numbers[i];
        }
    }
}
int main() {
    int numbers[] = {34, 21, 56, 78, 90, 23, 12};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    int min, max;
findMinMax(numbers, size, &min, &max);
printf("Minimum value: %d\n", min);
printf("Maximum value: %d\n", max);
    return 0;
}
```

OUTPUT:



```
Minimum value: 12
Maximum value: 90

--------------------------------
Process exited after 0.07009 seconds with return value 0
Press any key to continue . . .
```