Name: KUNAL KUMAR SAHOO

Roll No: 21BCP100

Subject: C-Programming Lab

Aim: To study and implement concepts of Loops in C.

Program 1: C program to print ODD numbers from 1 to N using while loop.

Code:

```
#include <stdio.h>

int main() {
   printf("Enter upper limit: ");
   int n; scanf("%d", &n);

   int i = 1;
   while (i <= n) {
      printf("%d\n", i);
      i += 2;
   }

   return 0;
}
```

Output:

```
Enter upper limit: 10

1

3

5

7

9
```

Program 2: C Program to print EVEN numbers from 1 to N using while loop.

Code:

```
#include <stdio.h>

int main() {
   printf("Enter upper limit: ");
   int n; scanf("%d", &n);

   int i = 2;
   while (i <= n) {
      printf("%d\n", i);
      i += 2;
   }

   return 0;
}
```

Output:

```
Enter upper limit: 10

2

4

6

8

10
```

Program 3: C program to print all uppercase alphabets using while loop.

Code:

```
#include <stdio.h>

int main() {
   int i = 0;
   while (i < 26) {
      printf("%c ", 'A'+i);
      i++;
   }

   return 0;
}
```

Output:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Program 4: C program to print all lowercase alphabets using while loop.

Code:

```
#include <stdio.h>

int main() {
   int i = 0;
   while (i < 26) {
      printf("%c ", 'a'+i);
      i++;
   }

   return 0;
}
```

Output:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Program 5: C Program to print numbers from 1 to N using while loop.

Code:

```c
#include <stdio.h>

int main() {
   printf("Enter upper limit: ");
   int n; scanf("%d", &n);

   int i = 1;
   while (i <= n) {
      printf("%d ", i);
      i++;
   }

   return 0;
}
```

Output:

```
Enter upper limit: 15

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

Program 6: C program to print 1 to 10 using while loop.

Code:

```c
#include <stdio.h>

int main() {
   int i = 1;
   while (i <= 10) {
      printf("%d ", i);
      i++;
   }

   return 0;
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10
```

Program 7: C Program to read a number and print its multiplication table.

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

int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   int i = 1;

   // While loop implementation:
   printf("While-loop implementation.\n");
   while (i <= 10) {
      printf("%d * %d = %d\n", n, i, n * i);
      i++;
   }

   // For loop implementation:
   printf("\nFor-loop implementation.\n");
   for (i = 1; i <= 10; i++)
      printf("%d * %d = %d\n", n, i, n*i);

   // Do-while loop implementation
   printf("\nDo-while-loop implementation.\n");
   i = 1;
   do {
      printf("%d * %d = %d\n", n, i, n * i);
      i++;
   } while(i <= 10);

   return 0;
}
```

Output:

```
Enter a number: 13
While-loop implementation.
13 * 1 = 13
13 * 2 = 26
13 * 3 = 39
13 * 4 = 52
13 * 5 = 65
13 * 6 = 78
13 * 7 = 91
13 * 8 = 104
13 * 9 = 117
13 * 10 = 130

For-loop implementation.
13 * 1 = 13
13 * 2 = 26
13 * 3 = 39
13 * 4 = 52
13 * 5 = 65
13 * 6 = 78
13 * 7 = 91
13 * 8 = 104
13 * 9 = 117
13 * 10 = 130

Do-while-loop implementation.
13 * 1 = 13
13 * 2 = 26
13 * 3 = 39
13 * 4 = 52
13 * 5 = 65
13 * 6 = 78
13 * 7 = 91
13 * 8 = 104
13 * 9 = 117
13 * 10 = 130
```

Program 8: C Program to print tables from 1 to 20.

Code:

```
#include <stdio.h>

int main() {
    int i = 1;
    while (i <= 20) {
        printf("Multiplication table of %d\n", i);
        for (int j = 1; j <= 10; j++)
            printf("%d * %d = %d\n", i, j, i*j);
        i++;
    }
    return 0;
}
```

Output:

```
Multiplication table of 1

1 * 1 = 1

1 * 2 = 2

1 * 3 = 3

1 * 4 = 4

1 * 5 = 5

1 * 6 = 6

1 * 7 = 7

1 * 8 = 8

1 * 9 = 9

1 * 10 = 10

Multiplication table of 2

2 * 1 = 2

2 * 2 = 4
```

```
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
Multiplication table of 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
Multiplication table of 4
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
```

```
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
Multiplication table of 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
Multiplication table of 6
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
```

```
6 * 9 = 54
6 * 10 = 60
Multiplication table of 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
Multiplication table of 8
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
Multiplication table of 9
```

```
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
Multiplication table of 10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
Multiplication table of 11
11 * 1 = 11
11 * 2 = 22
11 * 3 = 33
```

11 * 4 = 44

11 * 5 = 55

11 * 6 = 66

11 * 7 = 77

11 * 8 = 88

11 * 9 = 99

11 * 10 = 110

Multiplication table of 12

12 * 1 = 12

12 * 2 = 24

12 * 3 = 36

12 * 4 = 48

12 * 5 = 60

12 * 6 = 72

12 * 7 = 84

12 * 8 = 96

12 * 9 = 108

12 * 10 = 120

Multiplication table of 13

13 * 1 = 13

13 * 2 = 26

13 * 3 = 39

13 * 4 = 52

13 * 5 = 65

13 * 6 = 78

13 * 7 = 91

13 * 8 = 104

13 * 9 = 117

13 * 10 = 130

Multiplication table of 14

14 * 1 = 14

14 * 2 = 28

14 * 3 = 42

14 * 4 = 56

14 * 5 = 70

14 * 6 = 84

14 * 7 = 98

14 * 8 = 112

14 * 9 = 126

14 * 10 = 140

Multiplication table of 15

15 * 1 = 15

15 * 2 = 30

15 * 3 = 45

15 * 4 = 60

15 * 5 = 75

15 * 6 = 90

15 * 7 = 105

15 * 8 = 120

15 * 9 = 135

15 * 10 = 150

Multiplication table of 16

16 * 1 = 16

16 * 2 = 32

16 * 3 = 48

16 * 4 = 64

16 * 5 = 80

16 * 6 = 96

16 * 7 = 112

16 * 8 = 128

16 * 9 = 144

16 * 10 = 160

Multiplication table of 17

17 * 1 = 17

17 * 2 = 34

17 * 3 = 51

17 * 4 = 68

17 * 5 = 85

17 * 6 = 102

17 * 7 = 119

17 * 8 = 136

17 * 9 = 153

17 * 10 = 170

Multiplication table of 18

18 * 1 = 18

18 * 2 = 36

18 * 3 = 54

18 * 4 = 72

18 * 5 = 90

18 * 6 = 108

18 * 7 = 126

18 * 8 = 144

18 * 9 = 162

18 * 10 = 180

Multiplication table of 19

19 * 1 = 19

19 * 2 = 38

19 * 3 = 57

19 * 4 = 76

19 * 5 = 95

19 * 6 = 114

19 * 7 = 133

19 * 8 = 152

19 * 9 = 171

19 * 10 = 190

Multiplication table of 20

20 * 1 = 20

20 * 2 = 40

20 * 3 = 60

20 * 4 = 80

```
20 * 5 = 100

20 * 6 = 120

20 * 7 = 140

20 * 8 = 160

20 * 9 = 180

20 * 10 = 200
```

Program 9: C Program to check entered number is ZERO, POSITIVE or NEGATIVE until user does not want to quit.

Code:

```
#include <stdio.h>

int main() {
    char choice = 'n';
    do {
        printf("Enter a number: ");
        int n; scanf("%d", &n);
        if (n > 0) printf("POSITIVE\n");
        else if (n == 0) printf("ZERO\n");
        else printf("NEGATIVE\n");

        printf("Do you want to quit? (y/n): ");
        scanf("%s", &choice);
    } while(choice == 'n');
    return 0;
}
```

Output:

```
Enter a number: 0

ZERO

Do you want to quit? (y/n): n

Enter a number: 69

POSITIVE
```

```
Do you want to quit? (y/n): n

Enter a number: -234

NEGATIVE

Do you want to quit? (y/n): y
```

Program 10: C Program to print factorial of a number.

Code:

```c
#include <stdio.h>

int main() {
    printf("Enter a number: ");
    int n; scanf("%d", &n);
    if (n >= 0) {
        long factorial = 1;
        for (int i = 2; i <= n; i++)
            factorial *= i;
        printf("%d! = %ld\n", n, factorial);
    } else
        printf("Factorial of negative numbers don't exist!\n");

    return 0;
}
```

Output:

```
Enter a number: 6

6! = 720
```

Output:

```
Enter a number: 0

0! = 1
```

Output:

```
Enter a number: 1
```

```
1! = 1
```

Output:

```
Enter a number: -69
Factorial of negative numbers don't exist!
```

Program 11: C Program to find sum of first N natural number, N must be taken by the user.

Code:

```c
#include <stdio.h>
int main() {
   printf("Enter upper limit: ");
   int n; scanf("%d", &n);
   if (n <= 0)
      printf("Please enter a natural number.\n");
   else {
      long sum = 0;

      while (n > 0) {
         sum += n;
         n--;
      }
      printf("%ld\n", sum);
   }
   return 0;
}
```

Output:

```
Enter upper limit: 69
2415
```

Output:

```
Enter upper limit: 0
Please enter a natural number.
```

Program 12: C program to print all prime numbers from 1 to N.

Code:

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

int main() {
   printf("Enter upper limit: ");
   int n; scanf("%d", &n);

   int i = 2, count = 0;
   while (i <= n) {
      bool isPrime = true;
      for (int j = 2; j <= pow((double)i, 0.5); j++)
         if (i % j == 0) {
            isPrime = false;
            break;
         }
      if (isPrime) {
         printf("%d ", i);
         count++;
      }
      i++;
   }
   if (count == 0)
      printf("No prime numbers between 1 to %d\n", n);
   return 0;
}
```

Output:

```
Enter upper limit: 20

2 3 5 7 11 13 17 19
```

Program 13: C program to print all even and odd numbers between 1 to N.

Code:

```
#include <stdio.h>

int main() {
   printf("Enter upper limit: ");
   int n; scanf("%d", &n);

   printf("Odd numbers between 1 to %d:\n", n);
   int i = 1;
   while (i <= n) {
      printf("%d ", i);
      i += 2;
   }

   printf("\nEven numbers between 1 to %d:\n", n);
   for (i = 2; i <= n; i += 2)
      printf("%d ", i);

   printf("\n");
   return 0;
}
```

Output:

```
Enter upper limit: 30

Odd numbers between 1 to 30:

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29

Even numbers between 1 to 30:

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
```

Program 14: C program to print all Armstrong numbers from 1 to N.

Code:

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

int main() {
    printf("Enter upper limit: ");
    int n; scanf("%d", &n);

    int i = 1;
    while (i <= n) {
        int j = i, digits = 0;
        do {
            digits++;
            j /= 10;
        } while (j > 0);
        int sum = 0;
        for (j = i; j > 0; j/=10)
            sum += pow((double)(j % 10), digits);
        if (sum == i) printf("%d ", i);
        i++;
    }
    printf("\n");

    return 0;
}
```

Output:

```
Enter upper limit: 500

1 2 3 4 5 6 7 8 9 370 371 407
```

Program 15: C program to print square, cube and square root of all numbers from 1 to N.

Code:

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

int main() {
   printf("Enter upper limit: ");
   int n; scanf("%d", &n);

   if (n >= 1) {
      printf("Squares of numbers:\n");
      int i = 1;
      while (i <= n) {
         printf("%ld\n", (long)pow(i, 2));
         i++;
      }

      printf("\nCubes of numbers:\n");
      for (i = 1; i <= n; i++)
         printf("%ld\n", (long)pow(i, 3));

      printf("\nSquare roots of numbers:\n");
      i = 1;
      do {
         printf("%lf\n", sqrt(i));
         i++;
      } while(i <= n);
   }
   else
      printf("Enter natural numbers!\n");

   return 0;
}
```

Output:

```
Enter upper limit: 10
Squares of numbers:
1
4
9
16
24
36
49
64
81
99

Cubes of numbers:
1
8
27
64
124
216
343
512
729
1000

Square roots of numbers:
1.000000
1.414214
1.732051
2.000000
2.236068
2.449490
2.645751
2.828427
3.000000
3.162278
```

Program 16: C program to print all leap years from 1 to N.

Code:

```
#include <stdio.h>

int main() {
   printf("Enter upper limit year: ");
   int n; scanf("%d", &n);

   if (n >= 1) {
      for (int i = 1; i <= n; i++) {
         if ((i % 4 == 0) && (i % 100 != 0))
            printf("%d ", i);
         else if (i % 400 == 0)
            printf("%d ", i);
      }
   }
   else
      printf("Enter a year greater than or equal to 1.\n");

   return 0;
}
```

Output:

```
Enter a year: 2050
4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 104 108
112 116 120 124 128 132 136 140 144 148 152 156 160 164 168 172 176 180 184
188 192 196 204 208 212 216 220 224 228 232 236 240 244 248 252 256 260 264
268 272 276 280 284 288 292 296 304 308 312 316 320 324 328 332 336 340 344
348 352 356 360 364 368 372 376 380 384 388 392 396 400 404 408 412 416 420
424 428 432 436 440 444 448 452 456 460 464 468 472 476 480 484 488 492 496
504 508 512 516 520 524 528 532 536 540 544 548 552 556 560 564 568 572 576
580 584 588 592 596 604 608 612 616 620 624 628 632 636 640 644 648 652 656
660 664 668 672 676 680 684 688 692 696 704 708 712 716 720 724 728 732 736
740 744 748 752 756 760 764 768 772 776 780 784 788 792 796 800 804 808 812
816 820 824 828 832 836 840 844 848 852 856 860 864 868 872 876 880 884 888
892 896 904 908 912 916 920 924 928 932 936 940 944 948 952 956 960 964 968
972 976 980 984 988 992 996 1004 1008 1012 1016 1020 1024 1028 1032 1036
1040 1044 1048 1052 1056 1060 1064 1068 1072 1076 1080 1084 1088 1092
1096 1104 1108 1112 1116 1120 1124 1128 1132 1136 1140 1144 1148 1152
```

```
1156 1160 1164 1168 1172 1176 1180 1184 1188 1192 1196 1200 1204 1208
1212 1216 1220 1224 1228 1232 1236 1240 1244 1248 1252 1256 1260 1264
1268 1272 1276 1280 1284 1288 1292 1296 1304 1308 1312 1316 1320 1324
1328 1332 1336 1340 1344 1348 1352 1356 1360 1364 1368 1372 1376 1380
1384 1388 1392 1396 1404 1408 1412 1416 1420 1424 1428 1432 1436 1440
1444 1448 1452 1456 1460 1464 1468 1472 1476 1480 1484 1488 1492 1496
1504 1508 1512 1516 1520 1524 1528 1532 1536 1540 1544 1548 1552 1556
1560 1564 1568 1572 1576 1580 1584 1588 1592 1596 1600 1604 1608 1612
1616 1620 1624 1628 1632 1636 1640 1644 1648 1652 1656 1660 1664 1668
1672 1676 1680 1684 1688 1692 1696 1704 1708 1712 1716 1720 1724 1728
1732 1736 1740 1744 1748 1752 1756 1760 1764 1768 1772 1776 1780 1784
1788 1792 1796 1804 1808 1812 1816 1820 1824 1828 1832 1836 1840 1844
1848 1852 1856 1860 1864 1868 1872 1876 1880 1884 1888 1892 1896 1904
1908 1912 1916 1920 1924 1928 1932 1936 1940 1944 1948 1952 1956 1960
1964 1968 1972 1976 1980 1984 1988 1992 1996 2000 2004 2008 2012 2016
2020 2024 2028 2032 2036 2040 2044 2048
```

Program 17: C program to print all upper-case and lower-case alphabets.

Code:

```
#include <stdio.h>
int main() {
   int i = 0;
   printf("Uppercase alphabets:\n");
   while (i < 26) {
      printf("%c ", 'A'+i);
      i++;
   }
   printf("\nLowercase alphabets:\n");
   for (i = 0; i < 26; i++)
      printf("%c ", 'a'+i);
   printf("\n");
   return 0;
}
```

Output:

```
Uppercase alphabets:
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Lowercase alphabets:
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Program 18: C program to read age of 15 persons and count total Baby-age, School-age and Adult-age.

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

int main() {
   char choice = 'n';
   do {
      int kids = 0, students = 0, adults = 0;
      for (int i = 0; i < 15; i++) {
         printf("Enter person %d's age: ", i+1);
         int age; scanf("%d", &age);

         if (age < 0) printf("Enter some valid age\n");
         else if (age >= 0 && age < 8) kids++;
         else if (age >= 8 && age < 21) students++;
         else adults++;
      }
      printf("Number of kids: %d\n", kids);
      printf("Number of students: %d\n", students);
      printf("Number of adults: %d\n", adults);
      printf("Do you want to quit? (y/n): ");
      scanf("%s", &choice);
   } while (choice == 'n');
   return 0;
}
```

Output:

```
Enter person 1's age: 13
Enter person 2's age: 69
Enter person 3's age: 2
Enter person 4's age: 23
Enter person 5's age: 22
Enter person 6's age: 21
Enter person 7's age: 9
Enter person 8's age: 10
Enter person 9's age: 12
Enter person 10's age: 42
Enter person 11's age: 90
Enter person 12's age: 3
Enter person 13's age: 9
Enter person 14's age: 43
Enter person 15's age: 21
Number of kids: 2
Number of students: 5
Number of adults: 8
Do you want to quit? (y/n): n
Enter person 1's age: 21
Enter person 2's age: 32
Enter person 3's age: 5
Enter person 4's age: 6
Enter person 5's age: 12
Enter person 6's age: 45
Enter person 7's age: 12
Enter person 8's age: 13
Enter person 9's age: 14
Enter person 10's age: 60
Enter person 11's age: 10
Enter person 12's age: 32
Enter person 13's age: 09
Enter person 14's age: 13
Enter person 15's age: 4
Number of kids: 3
Number of students: 7
Number of adults: 5
Do you want to quit? (y/n): y
```

Aim: To study and implement concepts of Arrays in C.

Program 1: Write a program in C to store elements in an array and print it.

Code:

```
#include <stdio.h>

#define n 10

int main() {
   int arr[n], i;

   printf("Enter %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }

   printf("Elements in array are: ");
   for(i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}
```

Output:

```
Enter 10 elements in the array:
element - 0 : 1
element - 1 : 2
element - 2 : 3
element - 3 : 4
element - 4 : 5
element - 5 : 6
element - 6 : 7
element - 7 : 8
element - 8 : 9
element - 9 : 10
Elements in array are: 1 2 3 4 5 6 7 8 9 10
```

Program 2: Write a program in C to read n number of values in an array and display it in reverse order.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the number of elements to store in the array: ");
   int n; scanf("%d", &n);

   int arr[n], i;

   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }

   printf("The values stored into the array are:\n");
   for(i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }

   printf("\nThe values store into the array in reverse are:\n");
   while(--i >= 0) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}
```

Output:

```
Input the number of elements to store in the array: 3

Input 3 elements in the array:

element - 0 : 2

element - 1 : 5

element - 2 : 7

The values stored into the array are:

2 5 7

The values store into the array in reverse are:

7 5 2
```

Program 3: Write a program in C to find the sum of all elements of the array.

Code:

```c
#include <stdio.h>
int main() {
   printf("Input the number of elements to be stored in the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   long sum = 0;
   while(--i >= 0) {
      sum += arr[i];
   }

   printf("Sum of all elements stored in the array is: %ld\n", sum);

   return 0;
}
```

Output:

```
Input the number of elements to be stored in the array: 3

Input 3 elements in the array:

element - 0 : 2

element - 1 : 5

element - 2 : 8

Sum of all elements stored in the array is: 15
```

Program 4: Write a program in C to copy the elements of one array into another array.

Code:

```c
#include <stdio.h>
int main() {
   printf("Input the number of elements to be stored in the array: ");
   int n; scanf("%d", &n);
   int arr1[n], arr2[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr1[i]);
      arr2[i] = arr1[i];
   }
   printf("The elements stored in the first array are:\n");
   for(i = 0; i < n; i++) {
      printf("%d ", arr1[i]);
   }
   printf("\nThe elements stored in the second array are:\n");
   for(i = 0; i < n; i++) {
      printf("%d ", arr2[i]);
   }
   printf("\n");
   return 0;
}
```

Output:

```
Input the number of elements to be stored in the array: 3
Input 3 elements in the array:
element - 0 : 15
element - 1 : 10
element - 2 : 12
The elements stored in the first array are:
15 10 12
The elements stored in the second array are:
15 10 12
```

Program 5: Write a program in C to count a total number of duplicate elements in an array.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the number of elements to be stored in the array: ");
   int n; scanf("%d", &n);

   int arr[n], i;

   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }

   int unique[n], j, k = 1, c = 0;
   unique[0] = arr[0];
   for(i = 1; i < n; i++) {
      for(j = 0; j < k; j++) {
         if(arr[i] == unique[j]) {
            c++;
         }
         else {
            if(j == k-1) {
               unique[k] = arr[i];
               k++;
               break;
            }
         }
      }
   }
   printf("Total number of duplicate elements found in the array is: %d\n",
c);

   return 0;
}
```

Output:

```
Input the number of elements to be stored in the array: 3
Input 3 elements in the array:
element - 0 : 5
element - 1 : 1
element - 2 : 1
Total number of duplicate elements found in the array is: 1
```

Program 6: Write a program in C to print all unique elements in an array.

Code:

```
#include <stdio.h>

int main() {
    printf("Input the number of elements to be stored in the array: ");
    int n; scanf("%d", &n);

    int arr[n], i, j, c;

    printf("Input %d elements in the array:\n", n);
    for(i = 0; i < n; i++) {
        printf("element - %d : ", i);
        scanf("%d", &arr[i]);
    }

    printf("The unique elements found in the array are:\n");
    for(i = 0; i < n; i++) {
        c = 0;
        for(j = 0; j < n; j++) {
            if( i != j) {
                if(arr[i] == arr[j]) {
                    c++;
                }
            }
        }
        if(!c) {
            printf("%d ", arr[i]);
        }
    }
    printf("\n");

    return 0;
}
```

Output:

```
Input the number of elements to be stored in the array: 4

Input 4 elements in the array:

element - 0 : 3

element - 1 : 2

element - 2 : 2

element - 3 : 5

The unique elements found in the array are:

3 5
```

Program 7: Write a program in C to merge two arrays of same size swapped in descending order.

Code:

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

int main() {
   printf("Input the number of elements to be stored in the first array: ");
   int n1; scanf("%d", &n1); int arr1[n1], i;
   printf("Input %d elements in the array:\n", n1);
   for(i = 0; i < n1; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr1[i]);
   }

   printf("Input the number of elements to be stored in the second array: ");
   int n2; scanf("%d", &n2); int arr2[n2], j;
   printf("Input %d elements in the array:\n", n2);
   for(j = 0; j < n2; j++) {
      printf("element - %d : ", j);
      scanf("%d", &arr2[j]);
   }
```

```
    int arr3[n1+n2], temp;
    bool swapped;

    for(i = 0; i < n1; i++) {
        arr3[i] = arr1[i];
    }
    for(j = 0; j < n2; j++) {
        arr3[n1+j] = arr2[j];
    }

    for(i = 0; i < n1+n2-1; i++) {
        swapped = false;
        for(j = 1; j < n1+n2-i; j++) {
            if(arr3[j-1] < arr3[j]) {
                temp = arr3[j-1];
                arr3[j-1] = arr3[j];
                arr3[j] = temp;
                swapped = true;
            }
        }
        if(!swapped) {
            break;
        }
    }

    printf("The merged array in descending order is:\n");
    for(i = 0; i < n1+n2; i++) {
        printf("%d ", arr3[i]);
    }
    printf("\n");

    return 0;
}
```

Output:

```
Input the number of elements to be stored in the first array: 3
Input 3 elements in the array:
element - 0 : 1
element - 1 : 2
element - 2 : 3
Input the number of elements to be stored in the second array: 3
Input 3 elements in the array:
element - 0 : 2
element - 1 : 1
element - 2 : 3
The merged array in descending order is:
3 3 2 2 1 1
```

Program 8: Write a program in C to count the frequency of each element of an array.

Code:

```
#include <stdio.h>
int main() {
   printf("Input the number of elements to be stored in the array: ");
   int n; scanf("%d", &n);
   int arr[n], freq[n], i, j, c;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   printf("The frequency of all elements of an array:\n");
   for(i = 0; i < n; i++) {
      c = 1;
      for(j = i+1; j < n; j++) {
         if(arr[i] == arr[j]) {
            c++;
            freq[j] = -1;
         }
      }
      if(freq[i] != -1) {
         printf("%d occurs %d times\n", arr[i], c);
      }
   }

   return 0;
}
```

Output:

```
Input the number of elements to be stored in the array: 4
Input 4 elements in the array:
element - 0 : 25
element - 1 : 12
element - 2 : 25
element - 3 : 43
The frequency of all elements of an array:
25 occurs 2 times
12 occurs 1 times
43 occurs 1 times
```

Program 9: Write a program in C to find the maximum and minimum element in an array.

Code:

```
#include <stdio.h>
int main() {
   printf("Input the number of elements to be stored in the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   int max = arr[0], min = arr[0];
   while(--i) {
      if(max < arr[i]) {
         max = arr[i];
      }
      if(min > arr[i]) {
         min = arr[i];
      }
   }
   printf("Maximum element is: %d\n", max);
   printf("Minimum element is: %d\n", min);

   return 0;
}
```

Output:

```
Input the number of elements to be stored in the array: 4
Input 4 elements in the array:
element - 0 : 45
element - 1 : 69
element - 2 : 21
element - 3 : 23
Maximum element is: 69
Minimum element is: 21
```

Program 10: Write a program in C to separate odd and even integers in separate arrays.

Code:

```
#include <stdio.h>
int main() {
   printf("Input the number of elements to be stored in the array: ");
   int n; scanf("%d", &n);
   int odd[n], even[n], i, j, k, inp;
   printf("Input %d elements in the array:\n", n);
   j = 0; k = 0;
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &inp);
      if(inp % 2) {
         odd[j] = inp;
         j++;
      }
      else {
         even[k] = inp;
         k++;
      }
   }
   printf("Even elements are:\n");
   for(i = 0; i < k; i++) {
      printf("%d ", even[i]);
   }
   printf("\nOdd elements are:\n");
   for(i = 0; i < j; i++) {
      printf("%d ", odd[i]);
   }
   printf("\n");
   return 0;
}
```

Output:

```
Input the number of elements to be stored in the array: 5
Input 5 elements in the array:
element - 0 : 25
element - 1 : 47
element - 2 : 42
element - 3 : 56
element - 4 : 69
Even elements are:
42 56
Odd elements are:
25 47 69
```

Program 11: Write a program in C to sort elements of array in ascending order.

Code:

```
#include <stdio.h>
#include <stdbool.h>
int main() {
   printf("Input the size of the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   int j, temp; bool swapped;
   for(i = 0; i < n-1; i++) {
      swapped = false;
      for(j = 1; j < n-i; j++) {
         if(arr[j] < arr[j-1]) {
            temp = arr[j];
            arr[j] = arr[j-1];
            arr[j-1] = temp;
            swapped = true;
         }
      }
      if(!swapped) {
         break;
      }
   }
   printf("Elements of array in sorted ascending order:\n");
   for(i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
   return 0;
}
```

Output:

```
Input the size of the array: 5

Input 5 elements in the array:

element - 0 : 2

element - 1 : 7

element - 2 : 4

element - 3 : 5

element - 4 : 9

Elements of array in sorted ascending order:

2 4 5 7 9
```

Program 12: Write a program in C to sort elements of the array in descending order.

Code:

```
#include <stdio.h>
#include <stdbool.h>
int main() {
   printf("Input the size of the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   int j, temp; bool swapped;
   for(i = 0; i < n-1; i++) {
      swapped = false;
      for(j = 1; j < n-i; j++) {
         if(arr[j] > arr[j-1]) {
            temp = arr[j];
            arr[j] = arr[j-1];
            arr[j-1] = temp;
            swapped = true;
         }
      }
      if(!swapped) {
         break;
      }
   }
   printf("Elements of array in sorted descending order:\n");
   for(i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
   return 0;
}
```

Output:

```
Input the size of the array: 5
Input 5 elements in the array:
element - 0 : 2
element - 1 : 7
element - 2 : 4
element - 3 : 5
element - 4 : 9
Elements of array in sorted descending order:
9 7 5 4 2
```

Program 13: Write a program in C to insert New value in the array (sorted list)

Code:

```
#include <stdio.h>

#define MAX_SIZE 100

int main() {
   int n, arr[MAX_SIZE], i, target, target_idx;
   printf("Input the size of the array: ");
   scanf("%d", &n);

   printf("Input %d elements in the array in ascending order:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }

   printf("Input the value to be inserted: ");
   scanf("%d", &target);

   printf("The existing array list is:\n");
   for(i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }

   int start = 0, end = n-1, mid;
   while(start <= end) {
      mid = (start + end) / 2;
      if(arr[mid] == target) {
         target_idx = mid;
         break;
      }
      else if(arr[mid] < target) {
         target_idx = mid+1;
         start = mid+1;
      }
      else end = mid-1;
   }
```

```
    for(i = n-1; i >= target_idx; i--) {
        arr[i+1] = arr[i];
    }
    arr[target_idx] = target;

    printf("\nThe array after adding new value:\n");
    for(i = 0; i <= n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Output:

```
Input the size of the array: 5

Input 5 elements in the array in ascending order:

element - 0 : 2

element - 1 : 5

element - 2 : 7

element - 3 : 9

element - 4 : 11

Input the value to be inserted: 8

The existing array list is:

2 5 7 9 11

The array after adding new value:

2 5 7 8 9 11
```

Program 14: Write a program in C to insert New value in the array (unsorted list).

Code:

```
#include <stdio.h>

#define MAX_SIZE 100

int main() {
   int n, arr[MAX_SIZE], target, target_idx, i;

   printf("Input size of the array: ");
   scanf("%d", &n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   printf("Input the value to be inserted: ");
   scanf("%d", &target);
   printf("Input the position, where the value is to be inserted: ");
   scanf("%d", &target_idx); target_idx--;

   printf("The current list of the array:\n");
   for(i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   for(i = n-1; i >= target_idx; i--) {
      arr[i+1] = arr[i];
   }
   arr[target_idx] = target;
   printf("\nThe new list of array after insertion:\n");
   for(i = 0; i <= n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}
```

Output:

```
Input size of the array: 4
element - 0 : 1
element - 1 : 8
element - 2 : 7
element - 3 : 10
Input the value to be inserted: 5
Input the position, where the value is to be inserted: 2
The current list of the array:
1 8 7 10
The new list of array after insertion:
1 5 8 7 10
```

Program 15: Write a program in C to delete an element at desired position from an array.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the size of the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array in ascending order:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   printf("Input the position where to delete: ");
   int pos; scanf("%d", &pos); pos--;
   for(i = pos; i < n-1; i++) {
      arr[i] = arr[i+1];
   }
   printf("The new list is: ");
   for(i = 0; i < n-1; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
   return 0;
}
```

Output:

```
Input 5 elements in the array in ascending order:

element - 0 : 1

element - 1 : 2

element - 2 : 3

element - 3 : 4

element - 4 : 5

Input the position where to delete: 3

The new list is: 1 2 4 5
```

Program 16: Write a program in C to find the second largest element in an array.

Code:

```
#include <stdio.h>
#include <limits.h>
int main() {
   printf("Input the size of the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   int max = INT_MIN, secmax = INT_MIN;
   for(i = 0; i < n; i++) {
      if(arr[i] > max) {
         secmax = max;
         max = arr[i];
      }
      else {
         if(secmax < arr[i]) {
            secmax = arr[i];
         }
      }
   }
   printf("The second largest element in the array is: %d\n", secmax);
   return 0;
}
```

Output:

```
Input the size of the array: 5
Input 5 elements in the array:
element - 0 : 2
element - 1 : 9
element - 2 : 1
element - 3 : 4
element - 4 : 6
The second largest element in the array is: 6
```

Program 17: Write a program in C to find the second smallest element in an array.

Code:

```
#include <stdio.h>
#include <limits.h>
int main() {
   printf("Input the size of the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   int min = INT_MAX, secmin = INT_MAX;
   for(i = 0; i < n; i++) {
      if(arr[i] < min) {
         secmin = min;
         min = arr[i];
      }
      else {
         if(secmin > arr[i]) {
            secmin = arr[i];
         }
      }
   }
   printf("The second smallest element in the array is: %d\n", secmin);
   return 0;
}
```

Output:

```
Input the size of the array: 5
Input 5 elements in the array:
element - 0 : 2
element - 1 : 9
element - 2 : 1
element - 3 : 4
element - 4 : 6
The second smallest element in the array is: 2
```

Program 18: Write a program in C for a 2D array of size 3x3 and print the matrix.

Code:

```
#include <stdio.h>

#define rows 3
#define cols 3

int main() {
   int arr[rows][cols], row, col;

   printf("Input elements in the matrix:\n");
   for(row = 0; row < rows; row++) {
      for(col = 0; col < cols; col++) {
         printf("element - [%d],[%d] : ", row, col);
         scanf("%d", &arr[row][col]);
      }
   }

   printf("The matrix is:\n");
   for(row = 0; row < rows; row++) {
      for(col = 0; col < cols; col++) {
         printf("%d ", arr[row][col]);
      }
      printf("\n");
   }

   return 0;
}
```

Output:

```
Input elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [0],[2] : 3
element - [1],[0] : 4
element - [1],[1] : 5
element - [1],[2] : 6
element - [2],[0] : 7
element - [2],[1] : 8
element - [2],[2] : 9
The matrix is:
1 2 3
4 5 6
7 8 9
```

Program 19: Write a program in C for addition of two Matrices of same size.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the size of the square matrix (less than 5): ");
   int n; scanf("%d", &n);
   int A[n][n], B[n][n], i, j;
   printf("Input elements in the first matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &A[i][j]);
      }
   }
   printf("Input elements in the second matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &B[i][j]);
      }
   }
   printf("The first matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", A[i][j]);
      }
      printf("\n");
   }
   printf("The second matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", B[i][j]);
      }
      printf("\n");
   }
```

```
    printf("The addition of two matrices is:\n");
    for(i = 0; i < n; i++) {
        for(j = 0; j < n; j++) {
            printf("%d ", A[i][j] + B[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:

```
element - [0],[0] : 1

element - [0],[1] : 2

element - [1],[0] : 3

element - [1],[1] : 4

Input elements in the second matrix:

element - [0],[0] : 5

element - [0],[1] : 6

element - [1],[0] : 7

element - [1],[1] : 8

The first matrix is:

1 2

3 4

The second matrix is:

5 6

7 8

The addition of two matrices is:

6 8

10 12
```

Program 20: Write a program in C for subtraction of two Matrices.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the size of the square matrix (less than 5): ");
   int n; scanf("%d", &n);

   int A[n][n], B[n][n], i, j;

   printf("Input elements in the first matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &A[i][j]);
      }
   }

   printf("Input elements in the second matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &B[i][j]);
      }
   }
   printf("The first matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", A[i][j]);
      }
      printf("\n");
   }
   printf("The second matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", B[i][j]);
      }
      printf("\n");
   }
```

```
    printf("The subtraction of two matrices is:\n");
    for(i = 0; i < n; i++) {
        for(j = 0; j < n; j++) {
            printf("%d ", A[i][j] - B[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
Input the size of the square matrix (less than 5): 2
Input elements in the first matrix:
element - [0],[0] : 5
element - [0],[1] : 6
element - [1],[0] : 7
element - [1],[1] : 8
Input elements in the second matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [1],[0] : 3
element - [1],[1] : 4
The first matrix is:
5 6
7 8
The second matrix is:
1 2
3 4
The subtraction of two matrices is:
4 4
4 4
```

Program 21: Write a program in C for multiplication of two square Matrices.

Code:

```
#include <stdio.h>

int main() {
   int r1, r2, c1, c2;
   printf("Input the rows and columns of first matrix: ");
   scanf("%d %d", &r1, &c1);

   printf("Input the rows and columns of second matrix: ");
   scanf("%d %d", &r2, &c2);

   if(c1 == r2) {
      int a[r1][c1], b[r2][c2], i, j, k;

      printf("Input the elements in the first matrix:\n");
      for(i = 0; i < r1; i++) {
         for (j = 0; j < c1; j++) {
            printf("element - [%d],[%d] : ", i, j);
            scanf("%d", &a[i][j]);
         }
      }

      printf("Input the elements in the second matrix:\n");
      for(i = 0; i < r2; i++) {
         for (j = 0; j < c2; j++) {
            printf("element - [%d],[%d] : ", i, j);
            scanf("%d", &b[i][j]);
         }
      }

      printf("The first matrix is:\n");
      for(i = 0; i < r1; i++) {
         for(j = 0; j < c1; j++) {
            printf("%d ", a[i][j]);
         }
         printf("\n");
      }
```

```
        printf("The second matrix is:\n");
        for(i = 0; i < r2; i++) {
            for(j = 0; j < c2; j++) {
                printf("%d ", b[i][j]);
            }
            printf("\n");
        }

        printf("The multiplication of two matrices is:\n");
        int sum;
        for(i = 0; i < r1; i++) {
            for(j = 0; j < c2; j++) {
                sum = 0;
                for(k = 0; k < r2; k++) {
                    sum += a[i][k] * b[k][j];
                }
                printf("%d ", sum);
            }
            printf("\n");
        }
    }

    else {
        printf("Multplication of matrices of given dimensions is not possible.\n");
    }

    return 0;
}
```

Output:

```
Input the rows and columns of first matrix: 2 2

Input the rows and columns of second matrix: 2 2

Input the elements in the first matrix:

element - [0],[0] : 1

element - [0],[1] : 2

element - [1],[0] : 3

element - [1],[1] : 4
```

```
Input the elements in the second matrix:
element - [0],[0] : 5
element - [0],[1] : 6
element - [1],[0] : 7
element - [1],[1] : 8
The first matrix is:
1 2
3 4
The second matrix is:
5 6
7 8
The multiplication of two matrices is:
19 22
43 50
```

Program 22: Write a program in C to find transpose of a given matrix.

Code:

```
#include <stdio.h>

int main() {
	printf("Input the rows and columns of the matrix: ");
	int rows, cols; scanf("%d %d", &rows, &cols);
	int arr[rows][cols], t_arr[cols][rows], i, j;

	printf("Input the elements in the matrix:\n");
	for(i = 0; i < rows; i++) {
		for(j = 0; j < cols; j++) {
			printf("element - [%d],[%d] : ", i, j);
			scanf("%d", &arr[i][j]);
		}
	}
	printf("The matrix is:\n");
	for(i = 0; i < rows; i++) {
		for(j = 0; j < cols; j++) {
			printf("%d ", arr[i][j]);
		}
		printf("\n");
	}
	for(i = 0; i < rows; i++) {
		for(j = 0; j < cols; j++) {
			t_arr[j][i] = arr[i][j];
		}
	}

	printf("The transpose of the matrix is:\n");
	for(i = 0; i < cols; i++) {
		for(j = 0; j < rows; j++) {
			printf("%d ", t_arr[i][j]);
		}
		printf("\n");
	}

	return 0;
}
```

Output:

```
Input the rows and columns of the matrix: 3 3
Input the elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [0],[2] : 3
element - [1],[0] : 4
element - [1],[1] : 5
element - [1],[2] : 6
element - [2],[0] : 7
element - [2],[1] : 8
element - [2],[2] : 9
The matrix is:
1 2 3
4 5 6
7 8 9
The transpose of the matrix is:
1 4 7
2 5 8
3 6 9
```

Program 23: Write a program in C to find sum of right diagonals of a matrix.

Code:

```
#include <stdio.h>
int main() {
   printf("Input the size of the square matrix: ");
   int n; scanf("%d", &n);
   int arr[n][n], i, j;
   printf("Input elements in the matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &arr[i][j]);
      }
   }
   printf("The matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }
   long sum = 0;
   for(i = 0; i < n; i++)
      sum += arr[i][i];
   printf("Addition of the right diagonal elements is: %ld\n", sum);
   return 0;
}
```

Output:

```
Input the size of the square matrix: 2
Input elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [1],[0] : 3
element - [1],[1] : 4
The matrix is:
1 2
3 4
Addition of the right diagonal elements is: 5
```

Program 24: Write a program in C to find the sum of left diagonals of a matrix.

Code:

```
#include <stdio.h>
int main() {
   printf("Input the size of the square matrix: ");
   int n; scanf("%d", &n);
   int arr[n][n], i, j;
   printf("Input elements in the array:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &arr[i][j]);
      }
   }
   printf("The matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++)
         printf("%d ", arr[i][j]);
      printf("\n");
   }
   long sum = 0;
   for(i = n-1; i >= 0; i--)
      sum += arr[i][i];
   printf("Addition of the left diagonal elements is: %ld\n", sum);
   return 0;
}
```

Output:

```
Input the size of the square matrix: 2
Input elements in the array:
element - [0],[0] : 1
element - [0],[1] : 2
element - [1],[0] : 3
element - [1],[1] : 4
The matrix is:
1 2
3 4
Addition of the left diagonal elements is: 5
```

Program 25: Write a program in C to find sum of rows and columns of a Matrix.

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

int main() {
   printf("Input the size of the matrix: ");
   int n; scanf("%d", &n);

   int arr[n][n], i, j;

   printf("Input elements in the matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &arr[i][j]);
      }
   }

   printf("The matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }

   long sum;
   printf("The sum of rows and columns of the matrix is:\n");
   for(i = 0; i < n; i++) {
      sum = 0;
      for(j = 0; j < n; j++) {
         printf("%d ", arr[i][j]);
         sum += arr[i][j];
      }
      printf(" %ld\n", sum);
   }
   printf("\n");
```

```
    for(i = 0; i < n; i++) {
        sum = 0;
        for(j = 0; j < n; j++) {
            sum += arr[j][i];
        }
        printf("%ld ", sum);
    }
    printf("\n");

    return 0;
}
```

Output:

```
Input the size of the matrix: 2
Input elements in the matrix:
element - [0],[0] : 5
element - [0],[1] : 6
element - [1],[0] : 7
element - [1],[1] : 8
The matrix is:
5 6
7 8
The sum of rows and columns of the matrix is:
5 6  11
7 8  15

12 14
```

Program 26: Write a program in C to print or display the lower triangular of a given matrix.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the size of the square matrix: ");
   int n; scanf("%d", &n);
   int arr[n][n], i, j;

   printf("Input elements in the matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &arr[i][j]);
      }
   }
   printf("The matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }

   printf("Setting the lower matrix zero:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         if(i <= j) {
            printf("%d ", arr[i][j]);
         }
         else {
            printf("0 ");
         }
      }
      printf("\n");
   }

   return 0;
}
```

Output:

```
Input the size of the square matrix: 3
Input elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [0],[2] : 3
element - [1],[0] : 4
element - [1],[1] : 5
element - [1],[2] : 6
element - [2],[0] : 7
element - [2],[1] : 8
element - [2],[2] : 9
The matrix is:
1 2 3
4 5 6
7 8 9
Setting the lower matrix zero:
1 2 3
0 5 6
0 0 9
```

Program 27: Write a program in C to print or display the upper triangular of a given matrix.

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

int main() {
   printf("Input the size of the square matrix: ");
   int n; scanf("%d", &n);
   int arr[n][n], i, j;
   printf("Input elements in the matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &arr[i][j]);
      }
   }
   printf("The matrix is:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }
   printf("Setting the upper matrix zero:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         if(i >= j) {
            printf("%d ", arr[i][j]);
         }
         else {
            printf("0 ");
         }
      }
      printf("\n");
   }

   return 0;
}
```

Output:

```
Input the size of the square matrix: 3
Input elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [0],[2] : 3
element - [1],[0] : 4
element - [1],[1] : 5
element - [1],[2] : 6
element - [2],[0] : 7
element - [2],[1] : 8
element - [2],[2] : 9
The matrix is:
1 2 3
4 5 6
7 8 9
Setting the upper matrix zero:
1 0 0
4 5 0
7 8 9
```

Program 28: Write a program in C to calculate determinant of a 3 x 3 matrix.

Code:

```
#include <stdio.h>
int main() {
   int arr[3][3], i, j, det = 0;
   printf("Input elements in the matrix:\n");
   for(i = 0; i < 3; i++) {
      for(j = 0; j < 3; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &arr[i][j]);
      }
   }
   printf("The matrix is:\n");
   for(i = 0; i < 3; i++) {
      for(j = 0; j < 3; j++)
         printf("%d ", arr[i][j]);
      printf("\n");
   }
   for(i = 0; i < 3; i++) {
      det += (arr[0][i] * (arr[1][(i+1) % 3] * arr[2][(i+2) % 3] - arr[1][(i+2) %
3] * arr[2][(i+1) % 3]));
   }
   printf("The determinant of the matrix is: %d\n", det);
   return 0;
}
```

Output:

```
Input elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 0
element - [0],[2] : -1
element - [1],[0] : 0
element - [1],[1] : 0
element - [1],[2] : 1
element - [2],[0] : -1
element - [2],[1] : -1
element - [2],[2] : 0
The matrix is:
1 0 -1
0 0 1
-1 -1 0
```

Program 29: Write a program in C to accept a matrix and determine whether it is a sparse matrix.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the number of rows of the matrix: ");
   int rows; scanf("%d", &rows);
   printf("Input the number of columns of the matrix: ");
   int columns; scanf("%d", &columns);

   int arr[rows][columns], i, j, freq_zero = 0, freq_others = 0;

   printf("Input elements in the matrix:\n");
   for(i = 0; i < rows; i++) {
      for(j = 0; j < columns; j++) {
         printf("element - [%d],[%d] : ", i, j);
         scanf("%d", &arr[i][j]);

         if(arr[i][j] == 0) {
            freq_zero++;
         }
         else {
            freq_others++;
         }
      }
   }

   if(freq_zero > freq_others) {
      printf("The given matrix is sparse matrix.\n");
   }
   else {
      printf("The given matrix is not a sparse matrix.\n");
   }
   printf("There are %d number of zeros in the matrix.\n", freq_zero);

   return 0;
}
```

Output:

```
Input the number of rows of the matrix: 2
Input the number of columns of the matrix: 2
Input elements in the matrix:
element - [0],[0] : 0
element - [0],[1] : 0
element - [1],[0] : 1
element - [1],[1] : 0
The given matrix is sparse matrix.
There are 3 number of zeros in the matrix.
```

Program 30: Write a program in C to accept two matrices and check whether they are equal.

Code:

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

int main() {
   printf("Input rows and columns of the 1st matrix: ");
   int r1, c1; scanf("%d %d", &r1, &c1);
   printf("Input rows and columns of the 2nd matrix: ");
   int r2, c2; scanf("%d %d", &r2, &c2);

   if(r1 == r2 && c1 == c2) {
      printf("The matrices are comparable.\n");

      int a[r1][c1], b[r2][c2], i, j;

      printf("Input the elements in the first matrix:\n");
      for(i = 0; i < r1; i++) {
         for(j = 0; j < c1; j++) {
            printf("element - [%d],[%d] : ", i, j);
            scanf("%d", &a[i][j]);
         }
      }

      printf("Input the elements in the second matrix:\n");
      for(i = 0; i < r2; i++) {
         for(j = 0; j < c2; j++) {
            printf("element - [%d],[%d] : ", i, j);
            scanf("%d", &b[i][j]);
         }
      }

      printf("The first matrix is:\n");
      for(i = 0; i < r1; i++) {
         for(j = 0; j < c1; j++) {
            printf("%d ", a[i][j]);
         }
         printf("\n");
      }

      printf("The second matrix is:\n");
      for(i = 0; i < r2; i++) {
         for(j = 0; j < c2; j++) {
            printf("%d ", b[i][j]);
         }
         printf("\n");
      }

      bool answer = true;
      for(i = 0; i < r1; i++) {
         for(j = 0; j < c1; j++) {
            if(a[i][j] != b[i][j]) {
               answer = false;
               break;
            }
         }
      }
      if(answer) {
         printf("The two matrices are equal.\n");
      }
      else
         printf("The two matrices are not equal.\n");
   }
   else printf("The matrices are not comparable.\n");

   return 0;
}
```

Output:
```
Input rows and columns of the 1st matrix: 2 2
Input rows and columns of the 2nd matrix: 2 2
The matrices are comparable.
Input the elements in the first matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [1],[0] : 3
element - [1],[1] : 4
Input the elements in the second matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [1],[0] : 3
element - [1],[1] : 4
The first matrix is:
1 2
3 4
The second matrix is:
1 2
3 4
The two matrices are equal.
```

Program 31: Write a program in C to check whether a given matrix is an identity matrix.

Code:
```c
#include <stdio.h>
#include <stdbool.h>

int main() {
   printf("Input the number of rows of the matrix: ");
   int rows; scanf("%d", &rows);
   printf("Input the number of columns of the matrix: ");
   int columns; scanf("%d", &columns);

   if(rows == columns) {
      int arr[rows][columns], i, j;
      printf("Input elements in the matrix:\n");
```

```
    for(i = 0; i < rows; i++) {
        for(j = 0; j < columns; j++) {
            printf("element - [%d],[%d] : ", i, j);
            scanf("%d", &arr[i][j]);
        }
    }

    printf("The matrix is:\n");
    for(i = 0; i < rows; i++) {
        for(j = 0; j < columns; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }

    bool ans = true;
    for(i = 0; i < rows; i++) {
        for(j = 0; j < columns; j++) {
            if(i == j) {
                if(arr[i][j] != 1) {
                    ans = false;
                    break;
                }
            }
            else {
                if(arr[i][j] != 0) {
                    ans = false;
                    break;
                }
            }
        }
    }
    if(ans) {
        printf("The matrix is an identity matrix.\n");
    }
    else {
        printf("The matrix is not an identity matrix.\n");
    }

}
```

```
   else {
      printf("Only square matrices can be checked for identity matrices.\n");
   }
   return 0;
}
```

```
Output:
Input the number of rows of the matrix: 3
Input the number of columns of the matrix: 3
Input elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 0
element - [0],[2] : 0
element - [1],[0] : 0
element - [1],[1] : 1
element - [1],[2] : 0
element - [2],[0] : 0
element - [2],[1] : 0
element - [2],[2] : 1
The matrix is:
1 0 0
0 1 0
0 0 1
The matrix is an identity matrix.
```

```
Output:
Input the number of rows of the matrix: 2
Input the number of columns of the matrix: 2
Input elements in the matrix:
element - [0],[0] : 1
element - [0],[1] : 1
element - [1],[0] : 0
element - [1],[1] : 1
The matrix is:
1 1
0 1
The matrix is not an identity matrix.
```

Program 32: Write a program in C to find a pair with given sum in the array.

Code:

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

int main() {
   printf("Enter the size of the array: ");
   int n; scanf("%d", &n);

   int arr[n], i, j, sum;

   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }

   printf("Input the given sum: ");
   scanf("%d", &sum);

   bool ans = false;
   for(i = 0; i < n-1; i++) {
      for(j = i+1; j < n; j++) {
         if(arr[i] + arr[j] == sum) {
            ans = true;
            printf("Pair of elements can make the given sum by the value of
index %d and %d\n", i, j);
         }
      }
   }
   if(!ans) {
      printf("No such two elements found\n");
   }

   return 0;
}
```

```
Output:
Enter the size of the array: 6
Input 6 elements in the array:
element - 0 : 6
element - 1 : 8
element - 2 : 4
element - 3 : -5
element - 4 : 7
element - 5 : 9
Input the given sum: 15
Pair of elements can make the given sum by the value of index 0 and 5
Pair of elements can make the given sum by the value of index 1 and 4
```

Program 33: Write a program in C to find the majority element of an array.

Code:

```c
#include <stdio.h>
int main() {
   printf("Enter the size of the array: ");
   int n; scanf("%d", &n);
   int arr[n], i;
   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }
   /*
      Implementing: Moore's Voting Algorithm
      Reference: https://medium.com/@shahareg98/boyer-moore-voting-
algorithm-5b5f11580650
   */
   int majority_idx = 0, count = 1;
   for(i = 1; i < n; i++) {
      if(arr[majority_idx] == arr[i]) {
         count++;
      }
      else {
         count--;
      }
```

```
        if(count == 0) {
            majority_idx = i;
            count = 1;
        }
    }
    count = 0;
    for(i = 0; i < n; i++) {
        if(arr[i] == arr[majority_idx])
            count++;
    }
    if(count > n/2)
        printf("Majority element is: %d\n", arr[majority_idx]);
    else
        printf("There is no majority element in the array\n");
    return 0;
}
```

Output:

```
Enter the size of the array: 8
Input 8 elements in the array:
element - 0 : 4
element - 1 : 8
element - 2 : 4
element - 3 : 6
element - 4 : 7
element - 5 : 4
element - 6 : 4
element - 7 : 8
There is no majority element in the array
```

Output:

```
Enter the size of the array: 5
Input 5 elements in the array:
element - 0 : 1
element - 1 : 1
element - 2 : 2
element - 3 : 1
element - 4 : 2
Majority element is: 1
```

Program 34: Write a program in C to find the number occurring odd number of times in an array.

Code:

```
#include <stdio.h>

int main() {
   printf("Enter size of the array: ");
   int n; scanf("%d", &n);
   if(n % 2) {
      int arr[n], freq[n], i, j, count;
      printf("Input %d elements in the array:\n", n);
      for(i = 0; i < n; i++) {
         freq[i] = 0;
         printf("element - %d : ", i);
         scanf("%d", &arr[i]);
      }
      for(i = 0; i < n; i++) {
         count  = 1;
         for(j = i+1; j < n; j++) {
            if(arr[i] == arr[j]) {
               count++;
               freq[j] = -1;
            }
         }
         if(freq[i] != -1) {
            freq[i] = count;
            if(count % 2) {
               printf("The element odd number of times is: %d\n", arr[i]);
               break;
            }
         }
      }
   }
   else {
      printf("Either all elements have even frequencies or more than one 
numbers have odd frequencies\n");
   }
   return 0;
}
```

Output:

```
Enter size of the array: 9
Input 9 elements in the array:
element - 0 : 8
element - 1 : 3
element - 2 : 8
element - 3 : 5
element - 4 : 4
element - 5 : 3
element - 6 : 4
element - 7 : 3
element - 8 : 5
The element odd number of times is: 3
```

Program 35: Write a program in C to find the largest sum of contiguous subarray of an array.

Code:

```c
#include <stdio.h>

int main() {
   printf("Input the size of the array: ");
   int n; scanf("%d", &n);

   int arr[n], i, j, sum, max_sum;

   printf("Input %d elements in the array:\n", n);
   for(i = 0; i < n; i++) {
      printf("element - %d : ", i);
      scanf("%d", &arr[i]);
   }

   max_sum = arr[0];
   for(i = 0; i < n; i++) {
      sum = 0;
      for(j = i; j < n; j++) {
         sum += arr[j];
         if(sum > max_sum) {
            max_sum = sum;
            }
        }
    }

    printf("The largest sum of contiguous subarray is: %d\n", max_sum);

    return 0;
}
```

Output:

```
Input the size of the array: 9
Input 9 elements in the array:
element - 0 : 8
element - 1 : 3
element - 2 : 8
element - 3 : -5
element - 4 : 4
element - 5 : 3
element - 6 : -4
element - 7 : 3
element - 8 : 5
The largest sum of contiguous subarray is: 25
```

Program 36: Write a program in C to find the missing number from a given array. There are no duplicates in list.

Code:
```
#include <stdio.h>
#include <limits.h>

int main() {
        printf("Input the size of array: ");
        int n; scanf("%d", &n);

        int arr[n], i, max = INT_MIN, sum = 0;

        printf("Input %d elements in the array:\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr[i]);

                if(arr[i] > max) {
                        max = arr[i];
                }

                sum += arr[i];
        }

        int missing = max * (max + 1) / 2 - sum;

        if(missing) {
                printf("The missing number is: %d\n", missing);
        }
        else {
                printf("No number missing in between\n");
        }

        return 0;
}
```

Output:

```
Input the size of array: 8
Input 8 elements in the array:
element - 0 : 1
element - 1 : 3
element - 2 : 4
element - 3 : 2
element - 4 : 5
element - 5 : 6
element - 6 : 9
element - 7 : 8
The missing number is: 7
```

Program 37: Write a program in C to merge one sorted array into another sorted array.

Code:

```
#include <stdio.h>
#include <stdbool.h>
int main() {
        printf("Enter size of the bigger array: ");
        int n1; scanf("%d", &n1);
        printf("Enter size of the smaller array: ");
        int n2; scanf("%d", &n2);
        int arr1[n1], arr2[n2], arr3[n1+n2], i, j, tmp;
        printf("Input %d elements for the large array in ascending order:\n",
n1);
        for(i = 0; i < n1; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr1[i]);
        }
        printf("Input %d elements for the small array in ascending order:\n",
n2);
        for(i = 0; i < n2; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr2[i]);
        }
        for(i = 0; i < n1; i++) {
                arr3[i] = arr1[i];
        }
        for(i = 0; i < n2; i++) {
                arr3[n1+i] = arr2[i];
        }
        bool swapped;
        for(i = 0; i < n1+n2-1; i++) {
                swapped = false;
                for(j = 1; j < n1+n2-i; j++) {
                        if(arr3[j-1] > arr3[j]) {
                                tmp = arr3[j-1];
                                arr3[j-1] = arr3[j];
                                arr3[j] = tmp;
                                swapped = true;
                        }
                }
                if(!swapped) break;
        }
        printf("After merge, the new array is:\n");
        for(i = 0; i < n1+n2; i++) {
                printf("%d ", arr3[i]);
        }
        printf("\n");
        return 0;
}
```

Output:

```
Enter size of the bigger array: 7
Enter size of the smaller array: 6
Input 7 elements for the large array in ascending order:
element - 0 : 10
element - 1 : 12
element - 2 : 14
element - 3 : 16
element - 4 : 18
element - 5 : 20
element - 6 : 22
Input 6 elements for the small array in ascending order:
element - 0 : 11
element - 1 : 13
element - 2 : 15
element - 3 : 17
element - 4 : 19
element - 5 : 21
After merge, the new array is:
10 11 12 13 14 15 16 17 18 19 20 21 22
```

Program 38: Write a program in C to rotate an array by N positions.

Code:

```c
#include <stdio.h>

int main() {
	printf("Input the size of the array: ");
	int n; scanf("%d", &n);

	int arr[n], i, target_idx;

	printf("Input %d elements in the array:\n");
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}

	printf("Enter the position from where you want to rotate the array: ");
	scanf("%d", &target_idx);

	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}

	int temp[target_idx];

	printf("\nBefore the %d position, the values of the array are: ",
target_idx);
	for(i = 0; i < target_idx; i++) {
		temp[i] = arr[i];
		printf("%d ", temp[i]);
	}
```

```
        for(i = 0; i < n-target_idx; i++) {
                arr[i] = arr[target_idx+i];
        }

        for(i = n-target_idx; i < n; i++) {
                arr[i] = temp[i-n+target_idx];
        }

        printf("\nAfter rotating the array from %d position, the array is:\n",
target_idx);
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        printf("\n");

        return 0;
}
```

Output:

```
Input the size of the array: 12
Input 1 elements in the array:
element - 0 : 0
element - 1 : 3
element - 2 : 6
element - 3 : 9
element - 4 : 12
element - 5 : 14
element - 6 : 18
element - 7 : 19
element - 8 : 20
element - 9 : 22
element - 10 : 25
element - 11 : 27
Enter the position from where you want to rotate the array: 4
The given array is: 0 3 6 9 12 14 18 19 20 22 25 27
Before the 4 position, the values of the array are: 0 3 6 9
After rotating the array from 4 position, the array is:
12 14 18 19 20 22 25 27 0 3 6 9
```

Program 39: Write a program in C to print next greater elements in a given unsorted array. Elements for which no greater element exist, consider next greater element as -1.

Code:

```
#include <stdio.h>

int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], i, j;
	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	printf("Next Bigger Elements are:\n");
	for(i = 0; i < n; i++) {
		if(i < n-1) {
			for(j = i+1; j < n; j++) {
				if(arr[i] < arr[j]) {
					printf("Next bigger element of %d in the array 
is: %d\n", arr[i], arr[j]);
					break;
				}
			}
		}
		else {
			printf("Next bigger element of %d in the array is: -1\n", 
arr[i]);
		}
	}
	return 0;
}
```

Output:

```
Input the size of array: 6
Input 6 elements in the array:
element - 0 : 5
element - 1 : 3
element - 2 : 10
element - 3 : 9
element - 4 : 6
element - 5 : 13
The given array is: 5 3 10 9 6 13 Next Bigger Elements are:
Next bigger element of 5 in the array is: 10
Next bigger element of 3 in the array is: 10
Next bigger element of 10 in the array is: 13
Next bigger element of 9 in the array is: 13
Next bigger element of 6 in the array is: 13
Next bigger element of 13 in the array is: -1
```

Program 40: Write a program in C to find two elements whose sum is closest to zero.

Code:

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

int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);

	int arr[n], i, j, closest, sum, x, y;

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
```

```
        closest = arr[1]+arr[0];
        x = 0, y = 1;

        for(i = 0; i < n-1; i++) {
                for(j = i+1; j < n; j++) {
                        sum = arr[i] + arr[j];
                        if(abs(sum - 0) < abs(closest - 0)) {
                                closest = sum;
                                x = i, y = j;
                        }
                }
        }

        printf("\nThe pair of elements whose sum is minimum are:\n[%d, %d]\
n", arr[x], arr[y]);

        return 0;
}
```

Output:

```
Input the size of array: 10
Input 10 elements in the array:
element - 0 : 38
element - 1 : 44
element - 2 : 63
element - 3 : -51
element - 4 : -35
element - 5 : 19
element - 6 : 84
element - 7 : -69
element - 8 : 4
element - 9 : -46
The given array is: 38 44 63 -51 -35 19 84 -69 4 -46
The pair of elements whose sum is minimum are:
[44, -46]
```

Program 41: Write a program in C to find if a given integer x appears more than n/2 times in a sorted array of n integers.

Code:
```c
#include <stdio.h>

int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);

	int arr[n], i, target, cnt, target_cnt;

	if(n % 2) {
		target_cnt = (n-1) / 2;
	}
	else {
		target_cnt = n / 2;
	}

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}

	printf("Input the value: ");
	scanf("%d", &target);
	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}

	printf("\nThe given value is: %d\n", target);
	cnt = 0;
	while(--i >= 0) {
		if(arr[i] == target){
			cnt++;
		}
	}
```

```
        if(cnt > target_cnt) {
                printf("%d appears more than %d times in the given array.\n",
target, target_cnt);
        }
        else {
                printf("%d appears less than %d times in the given array.\n",
target, target_cnt);
        }

        return 0;
}
```

Output:

```
Input the size of array: 9
Input 9 elements in the array:
element - 0 : 1
element - 1 : 3
element - 2 : 3
element - 3 : 5
element - 4 : 4
element - 5 : 3
element - 6 : 2
element - 7 : 3
element - 8 : 3
Input the value: 4
The given array is: 1 3 3 5 4 3 2 3 3
The given value is: 4
4 appears less than 4 times in the given array.
```

Program 42: Write a program in C to check whether an array is subset of another array.

Code:
```
#include <stdio.h>
#include <stdbool.h>
int main() {
        printf("Input the size of larger array: ");
        int n1; scanf("%d", &n1);
        printf("Input the size of the smaller array: ");
        int n2; scanf("%d", &n2);
        int set[n1], sub_set[n2], i, j;
        printf("Input %d elements for the larger array:\n", n1);
        for(i = 0; i < n1; i++) {
                printf("element - %d : ", i);
                scanf("%d", &set[i]);
        }
        printf("Input %d elements for the smaller array:\n", n2);
        for(j = 0; j < n2; j++) {
                printf("element - %d : ", j);
                scanf("%d", &sub_set[j]);
        }
        printf("The given first array is: ");
        for(i = 0; i < n1; i++) {
                printf("%d ", set[i]);
        }
        printf("\nThe given second array is: ");
        for(j = 0; j < n2; j++) {
                printf("%d ", sub_set[j]);
        }
        bool ans, is_subset;
        for(i = 0; i < n2; i++) {
                ans = false;
                for(j = 0; j < n1; j++) {
                        if(sub_set[i] == set[j]) {
                                ans = true;
                                break;
                        }
                }
                if(!ans) break;
        }
        if(ans) {
                printf("\nThe second array is the subset of the first array.\n");
        }
        else {
                printf("\nThe second array is the subset of the first array.\n");
        }
        return 0;
}
```

```
Output:
Input the size of larger array: 9
Input the size of the smaller array: 5
Input 9 elements for the larger array:
element - 0 : 4
element - 1 : 8
element - 2 : 7
element - 3 : 11
element - 4 : 6
element - 5 : 9
element - 6 : 5
element - 7 : 0
element - 8 : 2
Input 5 elements for the smaller array:
element - 0 : 5
element - 1 : 4
element - 2 : 2
element - 3 : 0
element - 4 : 6
The given first array is: 4 8 7 11 6 9 5 0 2
The given second array is: 5 4 2 0 6
The second array is the subset of the first array.
```

Program 43: Write a program in C to move all zeroes to the end of a given array.

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

int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], i, idx;

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
		if(arr[i] != 0) {
			idx = i;
		}
	}

	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}

	for(i = 0; i < idx; i++) {
		if(arr[i] == 0) {
			arr[i] = arr[idx];
			arr[idx--] = 0;
		}
	}

	printf("\nThe new array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	printf("\n");
	return 0;
}
```

Objective:

```
Input the size of array: 10
Input 10 elements in the array:
element - 0 : 2
element - 1 : 5
element - 2 : 7
element - 3 : 0
element - 4 : 4
element - 5 : 0
element - 6 : 7
element - 7 : -5
element - 8 : 8
element - 9 : 0
The given array is: 2 5 7 0 4 0 7 -5 8 0
The new array is: 2 5 7 8 4 -5 7 0 0 0
```

Program 44: Write a program in C to find the row with maximum number of 1s.

Code:

```
#include <stdio.h>
int main() {
	printf("Input number of rows for 2-D array: ");
	int rows; scanf("%d", &rows);
	printf("Input number of columns for 2-D array: ");
	int cols; scanf("%d", &cols);
	int arr[rows][cols], i, j;
	printf("Input elements for the 2-D array:\n");
	for(i = 0; i < rows; i++) {
		for(j = 0; j < cols; j++) {
			printf("element - [%d],[%d] : ", i, j);
			scanf("%d", &arr[i][j]);
		}
	}
	printf("The given 2-D array is:\n");
	for(i = 0; i < rows; i++) {
		for(j = 0; j < rows; j++) {
			printf("%d ", arr[i][j]);
		}
		printf("\n");
```

```
        }
        int count = 0, max_count = 0, idx;
        for(i = 0; i < rows; i++) {
                count = 0;
                for(j = 0; j < cols; j++) {
                        if(arr[i][j] == 1) {
                                count++;
                        }
                }
                if(max_count < count) {
                        max_count = count;
                        idx = i;
                }
        }
        printf("The index of row with maximum 1s is: %d\n", idx);
        return 0;
}
```

Output:

```
Input number of rows for 2-D array: 5
Input number of columns for 2-D array: 5
Input elements for the 2-D array:
element - [0],[0] : 0
element - [0],[1] : 1
element - [0],[2] : 0
element - [0],[3] : 1
element - [0],[4] : 1
element - [1],[0] : 1
element - [1],[1] : 1
element - [1],[2] : 1
element - [1],[3] : 1
element - [1],[4] : 1
element - [2],[0] : 1
element - [2],[1] : 0
element - [2],[2] : 0
element - [2],[3] : 1
element - [2],[4] : 0
element - [3],[0] : 0
element - [3],[1] : 0
element - [3],[2] : 0
element - [3],[3] : 0
element - [3],[4] : 0
element - [4],[0] : 1
element - [4],[1] : 0
element - [4],[2] : 0
element - [4],[3] : 0
element - [4],[4] : 1
The given 2-D array is:
0 1 0 1 1
1 1 1 1 1
1 0 0 1 0
0 0 0 0 0
1 0 0 0 1
The index of row with maximum 1s is: 1
```

Program 45: Write a program in C to find maximum product subarray in a given array.

Code:

```c
#include <stdio.h>

int main() {
	printf("Input size of the array: ");
	int n; scanf("%d", &n);

	int arr[n], i, j, prod, max_prod;

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}

	max_prod = 1;

	for(i = 0; i < n-1; i++) {
		prod = 1;
		for(j = i; j < n; j++) {
			prod *= arr[j];
		}
		if(prod > max_prod) {
			max_prod = prod;
		}
	}

	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	printf("\nThe maximum product of a sub-array in the given array is:
%d\n", max_prod);

	return 0;
}
```

```
Output:
Input size of the array: 8
Input 8 elements in the array:
element - 0 : -4
element - 1 : 9
element - 2 : -7
element - 3 : 0
element - 4 : -15
element - 5 : 6
element - 6 : 2
element - 7 : -3
The given array is: -4 9 -7 0 -15 6 2 -3
The maximum product of a sub-array in the given array is: 540
```

Program 46: Write a program in C to replace every element with the greatest element on its right side.

Code:

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

int main() {

        printf("Input size of the array: ");
        int n; scanf("%d", &n);

        int arr[n], i, j, max, temp;

        printf("Input %d elements for the array:\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr[i]);
        }

        printf("The given array is: ");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        for(i = 0; i < n-1; i++) {
                max = INT_MIN;
                for(j = i+1; j < n; j++) {
                        if(arr[j] > max) {
                                max = arr[j];
                        }
                }
                arr[i] = max;
        }
        arr[n-1] = 0;
        printf("\nAfter replacements, the modified array is: ");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        printf("\n");
        return 0;
}
```

Output:

```
Input size of the array: 10
Input 10 elements for the array:
element - 0 : 7
element - 1 : 5
element - 2 : 8
element - 3 : 9
element - 4 : 6
element - 5 : 8
element - 6 : 5
element - 7 : 7
element - 8 : 4
element - 9 : 6
The given array is: 7 5 8 9 6 8 5 7 4 6
After replacements, the modified array is: 9 9 9 8 8 7 7 6 6 0
```

Program 47: Write a program in C to find the median of two sorted arrays of same size.

Code:

```
#include <stdio.h>

int main() {
        printf("Input the size of two arrays: ");
        int n; scanf("%d", &n);
        int arr1[n], arr2[n], i;
        float med1, med2;
        printf("Input %d elements in the first array:\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr1[i]);
        }
        printf("Input %d elements in the second array:\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr2[i]);
        }
        printf("The given array - 1 is: ");
        for(i = 0; i < n; i++) {
                printf("%d ", arr1[i]);
        }
        printf("\nThe given array - 2 is: ");
        for(i = 0; i < n; i++) {
                printf("%d ", arr2[i]);
        }

        if(n % 2) {
                med1 = arr1[(n+1)/2-1];
                med2 = arr2[(n+1)/2-1];
        }
        else {
                med1 = (arr1[n/2-1] + arr1[n/2]) / 2;
                med2 = (arr2[n/2-1] + arr2[n/2]) / 2;
        }
        printf("\nThe median of two sorted arrays is: %f\n",
(med1+med2)/2.0);
        return 0;
}
```

Output:

```
Input the size of two arrays: 5
Input 5 elements in the first array:
element - 0 : 1
element - 1 : 5
element - 2 : 13
element - 3 : 24
element - 4 : 35
Input 5 elements in the second array:
element - 0 : 3
element - 1 : 8
element - 2 : 15
element - 3 : 17
element - 4 : 32
The given array - 1 is: 1 5 13 24 35
The given array - 2 is: 3 8 15 17 32
The median of two sorted arrays is: 14.000000
```

Program 48: Write a program in C to find the product of an array such that product is equal to the product of all the elements of arr[] except arr[i].

Code:
```
#include <stdio.h>
int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], i, j, prod = 1;
	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
		prod *= arr[i];
	}
	printf("\nThe product array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", prod / arr[i]);
	}
	printf("\n");
	return 0;
}
```

Output:
```
Input the size of array: 6
Input 6 elements in the array:
element - 0 : 1
element - 1 : 2
element - 2 : 3
element - 3 : 4
element - 4 : 5
element - 5 : 6
The given array is: 1 2 3 4 5 6
The product array is: 720 360 240 180 144 120
```

Program 49: Write a program in C to search an element in a row wise and column wise sorted matrix.

Code:
```
#include <stdio.h>
#include <stdbool.h>

int main() {
	printf("Input number of rows of matrix: ");
	int rows; scanf("%d", &rows);

	printf("Input number of columns of matrix: ");
	int cols; scanf("%d", &cols);

	int arr[rows][cols], target, i, j;
	bool found = false;

	printf("Input elements in the matrix:\n");
	for(i = 0; i < rows; i++) {
		for(j = 0; j < cols; j++) {
			printf("element - [%d],[%d] : ", i, j);
			scanf("%d", &arr[i][j]);
		}
	}

	printf("Input target element: ");
	scanf("%d", &target);

	printf("The given array in the matrix form is:\n");
	for(i = 0; i < rows; i++) {
		for(j = 0; j < cols; j++) {
			printf("%d ", arr[i][j]);
		}
		printf("\n");
	}

	printf("The given value for searching is: %d\n", target);
```

```
        for(i = 0; i < rows; i++) {
                for(j = 0; j < cols; j++) {
                        if(arr[i][j] == target) {
                                printf("The element Found at the position in the
matrix is: %d, %d\n", i, j);
                                found = true;
                        }
                        if(arr[i][j] > target) break;
                }
        }
        if(!found) {
                printf("The element not found in the matrix\n");
        }

        return 0;
}
```

Output:

```
Input number of rows of matrix: 4
Input number of columns of matrix: 4
Input elements in the matrix:
element - [0],[0] : 15
element - [0],[1] : 23
element - [0],[2] : 31
element - [0],[3] : 39
element - [1],[0] : 18
element - [1],[1] : 26
element - [1],[2] : 36
element - [1],[3] : 43
element - [2],[0] : 25
element - [2],[1] : 28
element - [2],[2] : 37
element - [2],[3] : 48
element - [3],[0] : 30
element - [3],[1] : 34
element - [3],[2] : 39
element - [3],[3] : 50
Input target element: 37
```

```
The given array in the matrix form is:
15 23 31 39
18 26 36 43
25 28 37 48
30 34 39 50
The given value for searching is: 37
The element Found at the position in the matrix is: 2, 2
```

Program 50: Write a program in C to find all numbers that occur odd number of times in an array.

Code:

```c
#include <stdio.h>

int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], freq[n], i, j, count;

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
	printf("The given array is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	for(i = 0; i < n-1; i++) {
		count = 1;
		for(j = i+1; j < n; j++) {
			if(arr[i] == arr[j]) {
				freq[j] = -1;
				count++;
			}
		}
		if(freq[i] != -1) {
			freq[i] = count;
		}
	}
```

```
    printf("\nThe numbers occurring odd number of times are: ");
    for(i = 0; i < n; i++) {
        if(freq[i] != -1 && freq[i] % 2) {
            printf("%d ", arr[i]);
        }
    }
    printf("\n");

    return 0;
}
```

Output:

```
Input the size of array: 10
Input 10 elements in the array:
element - 0 : 6
element - 1 : 7
element - 2 : 3
element - 3 : 6
element - 4 : 8
element - 5 : 7
element - 6 : 6
element - 7 : 8
element - 8 : 3
element - 9 : 3
The given array is: 6 7 3 6 8 7 6 8 3 3
The numbers occurring odd number of times are: 6 3
```

Program 51: Write a program in C to find the median of two sorted arrays of different size.

Code:

```
#include <stdio.h>
int main() {
        printf("Input the size of array-1: ");
        int n1; scanf("%d", &n1);
        printf("Input the size of array-2: ");
        int n2; scanf("%d", &n2);
        int arr1[n1], arr2[n2], i;
        float med1, med2;
        printf("Input %d elements in the first array:\n", n1);
        for(i = 0; i < n1; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr1[i]);
        }
        printf("Input %d elements in the second array:\n", n2);
        for(i = 0; i < n2; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr2[i]);
        }
        printf("The given array - 1 is: ");
        for(i = 0; i < n1; i++) {
                printf("%d ", arr1[i]);
        }
        printf("\nThe given array - 2 is: ");
        for(i = 0; i < n2; i++) {
                printf("%d ", arr2[i]);
        }

        med1 = (n1 % 2 == 1) ? (float)arr1[(n1-1)/2] : (arr1[n1/2] +
arr1[n1/2+1])/2.0;
        med2 = (n2 % 2 == 1) ? (float)arr2[(n2-1)/2] : (arr2[n2/2] +
arr2[n2/2+1])/2.0;

        printf("\nThe median of two sorted arrays is: %f\n", (med1+med2)/2);

        return 0;
}
```

Output:

```
Input the size of array-1: 3
Input the size of array-2: 5
Input 3 elements in the first array:
element - 0 : 90
element - 1 : 240
element - 2 : 300
Input 5 elements in the second array:
element - 0 : 10
element - 1 : 13
element - 2 : 14
element - 3 : 20
element - 4 : 25
The given array - 1 is: 90 240 300
The given array - 2 is: 10 13 14 20 25
The median of two sorted arrays is: 127.000000
```

Program 52: Write a program in C to find the sum of upper triangular elements of a matrix.

Code:

```c
#include <stdio.h>

int main() {
	printf("Input order of square matrix: ");
	int n; scanf("%d", &n);

	int arr[n][n], i, j, sum = 0;

	printf("Input elements of the matrix:\n");
	for(i = 0; i < n; i++) {
		for(j = 0; j < n; j++) {
			printf("element - [%d],[%d] : ", i, j);
			scanf("%d", &arr[i][j]);
		}
	}

	printf("The given matrix is:\n");
```

```
        for(i = 0; i < n; i++) {
                for(j = 0; j < n; j++) {
                        printf("%d ", arr[i][j]);
                }
                printf("\n");
        }
        printf("The elements being summed of the upper triangular matrix are:
");
        for(i = 0; i < n; i++) {
                for(j = 0; j < n; j++) {
                        if(i < j) {
                                printf("%d ", arr[i][j]);
                                sum += arr[i][j];
                        }
                }
        }
        printf("\nThe sum of the upper triangular matrix elements is: %d\n",
sum);

        return 0;
}
```

Output:

```
Input order of square matrix: 3
Input elements of the matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [0],[2] : 3
element - [1],[0] : 4
element - [1],[1] : 5
element - [1],[2] : 6
element - [2],[0] : 7
element - [2],[1] : 8
element - [2],[2] : 9
The given matrix is:
1 2 3
4 5 6
7 8 9
The elements being summed of the upper triangular matrix are: 2 3 6
The sum of the upper triangular matrix elements is: 11
```

Program 53: Write a program in C to find the sum of lower triangular elements of a matrix.

Code:

```
#include <stdio.h>

int main() {
	printf("Input order of square matrix: ");
	int n; scanf("%d", &n);
	int arr[n][n], i, j, sum = 0;

	printf("Input elements of the matrix:\n");
	for(i = 0; i < n; i++) {
		for(j = 0; j < n; j++) {
			printf("element - [%d],[%d] : ", i, j);
			scanf("%d", &arr[i][j]);
		}
	}
	printf("The given matrix is:\n");
	for(i = 0; i < n; i++) {
		for(j = 0; j < n; j++) {
			printf("%d ", arr[i][j]);
		}
		printf("\n");
	}

	printf("The elements being summed of the lower triangular matrix are: 
");
	for(i = 0; i < n; i++) {
		for(j = 0; j < n; j++) {
			if(i > j) {
				printf("%d ", arr[i][j]);
				sum += arr[i][j];
			}
		}
	}
	printf("\nThe sum of the lower triangular matrix elements is: %d\n", 
sum);

	return 0;
}
```

Output:

```
Input order of square matrix: 3
Input elements of the matrix:
element - [0],[0] : 1
element - [0],[1] : 2
element - [0],[2] : 3
element - [1],[0] : 4
element - [1],[1] : 5
element - [1],[2] : 6
element - [2],[0] : 7
element - [2],[1] : 8
element - [2],[2] : 9
The given matrix is:
1 2 3
4 5 6
7 8 9
The elements being summed of the lower triangular matrix are: 4 7 8
The sum of the lower triangular matrix elements is: 19
```

Program 54: Write a program in C to generate a random permutation of array elements.

Code:

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

int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);

	int arr[n], i, j, tmp;

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
```

```
        printf("The given array is:\n");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }

        srand(time(0));
        for(i = n-1; i > 0; i--) {
                j = (rand() % (n));
                tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
        }

        printf("\nThe shuffled elements in the array are:\n");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        printf("\n");

        return 0;
}
```

```
Output:
Input the size of array: 5
Input 5 elements in the array:
element - 0 : 1
element - 1 : 2
element - 2 : 3
element - 3 : 4
element - 4 : 5
The given array is:
1 2 3 4 5
The shuffled elements in the array are:
5 1 4 3 2
```

Program 55: Write a program in C to find the maximum repeating number in a given array.

Code:

```
#include <stdio.h>

int main() {
	printf("Input the size of the array: ");
	int n; scanf("%d", &n);

	int arr[n], i, j, mode, freq, max_freq;

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}

	printf("The given array is:\n");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}

	mode = arr[0];
	freq = max_freq = 1;
	for(i = 1; i < n-1; i++) {
		for(j = i+1; j < n; j++) {
			if(arr[i] == arr[j]) {
				freq++;
			}
		}
		if(freq > max_freq) {
			max_freq = freq;
			mode = arr[i];
		}
	}

	printf("\nThe maximum repeating number is: %d\n", mode);

	return 0;
}
```

Output:

```
Input the size of the array: 11
Input 11 elements in the array:
element - 0 : 2
element - 1 : 3
element - 2 : 3
element - 3 : 5
element - 4 : 3
element - 5 : 4
element - 6 : 1
element - 7 : 7
element - 8 : 7
element - 9 : 7
element - 10 : 7
The given array is:
2 3 3 5 3 4 1 7 7 7 7
The maximum repeating number is: 7
```

Program 56: Write a program in C to find a pair with the given difference.

Code:

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

int main() {
        printf("Input the size of array: ");
        int n; scanf("%d", &n);

        int arr[n], diff, i, j;
        bool found;

        printf("Input %d elements in the array:\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr[i]);
        }
        printf("Input the difference value: ");
        scanf("%d", &diff);

        printf("The given array is:\n");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        printf("\nThe given difference is: %d\n", diff);

        found = false;
        for(i = 0; i < n-1; i++) {
                for(j = i+1; j < n; j++) {
                        if(arr[i] - arr[j] == diff ||
                                arr[j] - arr[i] == diff) {
                                printf("The pair is: (%d, %d)\n", arr[i], arr[j]);
                                found = true;
                        }
                }
        }

        if(!found) {
                printf("No such pairs found\n");
        }
        return 0;
}
```

Output:

```
Input the size of array: 5
Input 5 elements in the array:
element - 0 : 1
element - 1 : 15
element - 2 : 39
element - 3 : 75
element - 4 : 92
Input the difference value: 53
The given array is:
1 15 39 75 92
The given difference is: 53
The pair is: (39, 92)
```

Program 57: Write a program in C to find the minimum distance between two numbers in a given array.

Code:

```
#include <stdio.h>
#include <limits.h>
int main() {
        printf("Input the size of array: ");
        int n; scanf("%d", &n);
        int arr[n], i, j, num1, num2, min_dist;

        printf("Input %d elements in the array:\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr[i]);
        }

        printf("Input two numbers: ");
        scanf("%d %d", &num1, &num2);

        printf("The given array is:\n");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }

        min_dist = INT_MAX;

        for(i = 0; i < n-1; i++) {
                for(j = i+1; j < n; j++) {
                        if(arr[i] == num1 && arr[j] == num2) {
                                if(min_dist > j-i) {
                                        min_dist = j-i;
                                }
                        }
                }
        }
        printf("\nThe minimum distance between %d and %d is: %d\n",
num1, num2, min_dist);

        return 0;
}
```

Output:

```
Input the size of array: 10
Input 10 elements in the array:
element - 0 : 7
element - 1 : 9
element - 2 : 5
element - 3 : 11
element - 4 : 7
element - 5 : 4
element - 6 : 12
element - 7 : 6
element - 8 : 2
element - 9 : 14
Input two numbers: 7 11
The given array is:
7 9 5 11 7 4 12 6 2 14
The minimum distance between 7 and 11 is: 3
```

Program 58: Write a program in C that checks whether the elements in an array are sorted or not.

Code:

```
#include <stdio.h>
#include <stdbool.h>
int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], i;
	bool asc, desc;
	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++){
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
	printf("The given array is:\n");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
```

```
        for(i = 0; i < n-1; i++) {
                if(arr[i] < arr[i+1])
                        asc = true;
                else{
                        asc = false;
                        break;
                }
        }
        for(i = 0; i < n-1; i++) {
                if(arr[i] > arr[i+1])
                        desc = true;
                else{
                        desc = false;
                        break;
                }
        }

        if(asc || desc)
                printf("\nThe elements of the array are sorted.\n");
        else
                printf("\nThe elements of the array are not sorted.\n");

        return 0;
}
```

Output:

```
Input the size of array: 6
Input 6 elements in the array:
element - 0 : 7
element - 1 : 4
element - 2 : 3
element - 3 : 5
element - 4 : 6
element - 5 : 2
The given array is:
7 4 3 5 6 2
The elements of the array are not sorted.
```

Output:

```
Input the size of array: 5
Input 5 elements in the array:
element - 0 : 1
element - 1 : 2
element - 2 : 3
element - 3 : 4
element - 4 : 5
The given array is:
1 2 3 4 5
The elements of the array are sorted.
```

Program 59: Write a program in C to rearrange positive and negative numbers alternatively in a given array.

Code:

```
#include <stdio.h>
int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], i, j, temp;
	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
	printf("The given array is:\n");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	for(i = 0; i < n; i++) {
		if(arr[i] < 0) {
			for(j = i+2; j < n; j++) {
				if(arr[j] > 0) {
					temp = arr[i+1];
					arr[i+1] = arr[j];
					arr[j] = temp;
					break;
				}
```

```
                }
            }
            else {
                for(j = i+2; j < n; j++) {
                    if(arr[j] < 0) {
                        temp = arr[i+1];
                        arr[i+1] = arr[j];
                        arr[j] = temp;
                        break;
                    }
                }
            }
        }

        printf("\nThe re-arranged array is:\n");
        for(i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
        printf("\n");
        return 0;
}
```

```
Output:
Input the size of array: 11
Input 11 elements in the array:
element - 0 : -4
element - 1 : 8
element - 2 : -5
element - 3 : -6
element - 4 : 5
element - 5 : -9
element - 6 : 7
element - 7 : 1
element - 8 : -21
element - 9 : -11
element - 10 : 19
The given array is:
-4 8 -5 -6 5 -9 7 1 -21 -11 19
The re-arranged array is:
-4 5 -6 8 -9 7 -21 19 -11 1 -5
```

Program 60: Write a program in C to segregate 0s and 1s in an array.

Code:
```
#include <stdio.h>
int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], i, freq_zero = 0;
	printf("Input %d elements in the array (0s and 1s only):\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
		if(arr[i] != 0 && arr[i] != 1) {
			printf("Please enter either 0 or 1\n");
			i--;
		}
	}
	freq_zero = 0;
	printf("The given array is:\n");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
		if(arr[i] == 0) {
			freq_zero++;
		}
	}
	for(i = 0; i < n; i++) {
		if(i < freq_zero)
			arr[i] = 0;
		else
			arr[i] = 1;
	}
	printf("\nThe array after segregation is: ");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	printf("\n");
	return 0;
}
```

Output:
```
Input the size of array: 9
Input 9 elements in the array (0s and 1s only):
element - 0 : 1
element - 1 : 0
element - 2 : 1
element - 3 : 0
element - 4 : 0
element - 5 : 1
element - 6 : 0
element - 7 : 1
element - 8 : 1
The given array is:
1 0 1 0 0 1 0 1 1
The array after segregation is: 0 0 0 0 1 1 1 1 1
```

Program 61: Write a program in C to segregate even and odd elements on an array.

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

int main() {
	printf("Input the size of array: ");
	int n; scanf("%d", &n);
	int arr[n], i, j, temp;

	printf("Input %d elements in the array:\n", n);
	for(i = 0; i < n; i++) {
		printf("element - %d : ", i);
		scanf("%d", &arr[i]);
	}
	printf("The given array is:\n");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	for(i = 0; i < n; i++) {
		if(arr[i] % 2 == 1) {
			for(j = i+1; j < n; j++) {
				if(arr[j] % 2 == 0) {
```

```
                                        temp = arr[i];
                                        arr[i] = arr[j];
                                        arr[j] = temp;
                                        break;
                                }
                        }
                }
        }
        printf("\nThe array after segregation is: ");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        printf("\n");
        return 0;
}
```

```
Output:
Input the size of array: 9
Input 9 elements in the array:
element - 0 : 17
element - 1 : 42
element - 2 : 19
element - 3 : 7
element - 4 : 27
element - 5 : 24
element - 6 : 30
element - 7 : 54
element - 8 : 73
The given array is:
17 42 19 7 27 24 30 54 73
The array after segregation is: 42 24 30 54 27 17 19 7 73
```

Program 62: Write a program in C to rearrange an array in such an order that– smallest, largest, 2nd smallest, 2nd largest and on.

Code:
```
#include <stdio.h>
#include <limits.h>

int main() {
        printf("Input the size of array: ");
        int n; scanf("%d", &n);
        int arr[n], i, j, max, min, max_idx, min_idx;
        printf("Input %d elements in the array:\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr[i]);
        }
        printf("The given array is:\n");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        for(i = 0; i < n-1; i++) {
                max = INT_MIN, min = INT_MAX;
                for(j = i; j < n; j++) {
                        if(arr[j] > max) {
                                max = arr[j];
                                max_idx = j;
                        }
                        if(arr[j] < min) {
                                min = arr[j];
                                min_idx = j;
                        }
                }
                if(i % 2 == 0) {
                        arr[min_idx] = arr[i];
                        arr[i] = min;
                }
                else {
                        arr[max_idx] = arr[i];
                        arr[i] = max;
                }
```

```
	}
	printf("\nThe new array is:\n");
	for(i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	printf("\n");
	return 0;
}
```

Output:

```
Input the size of array: 9
Input 9 elements in the array:
element - 0 : 5
element - 1 : 8
element - 2 : 1
element - 3 : 4
element - 4 : 2
element - 5 : 9
element - 6 : 3
element - 7 : 7
element - 8 : 6
The given array is:
5 8 1 4 2 9 3 7 6
The new array is:
1 9 2 8 3 7 4 6 5
```

Program 63: Write a program in C to update every array element with multiplication of previous and next numbers in array.

Code:
```c
#include <stdio.h>

int main() {
        printf("Input the size of array: ");
        int n; scanf("%d", &n);

        int arr[n], prod[n], i;

        printf("Input %d elements in the array\n", n);
        for(i = 0; i < n; i++) {
                printf("element - %d : ", i);
                scanf("%d", &arr[i]);
        }

        printf("The given array is:\n");
        for(i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }

        prod[0] = arr[0] * arr[1];
        prod[n-1] = arr[n-1] * arr[n-2];

        for(i = 1; i < n-1; i++) {
                prod[i] = arr[i-1] * arr[i+1];
        }

        printf("\nThe new array is:\n");
        for(i = 0; i < n; i++) {
                printf("%d ", prod[i]);
        }
        printf("\n");

        return 0;
}
```

Objective:

```
Input the size of array: 6
Input 6 elements in the array
element - 0 : 1
element - 1 : 2
element - 2 : 3
element - 3 : 4
element - 4 : 5
element - 5 : 6
The given array is:
1 2 3 4 5 6
The new array is:
2 3 8 15 24 30
```

Aim: To study and implement concepts of Strings in C.

Program 1: Write a program in C to input a string and print it.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
	printf("Input the string: ");
	char str[SIZE];

	// Not using gets() because it is not supported in Linux.
	fgets(str, sizeof(str), stdin);

	printf("%s\n", str);

	return 0;
}
```

Output:

```
Input the string: Kunal Kumar Sahoo
Kunal Kumar Sahoo
```

Program 2: Write a program in C to find the length of a string without using library function.

Code:

```
#include <stdio.h>
#define SIZE 100
int main() {
	printf("Input the string: ");
	char str[SIZE];
	fgets(str, sizeof(str), stdin);
	int len = 0;
	while(str[len++] != '\0');
	len--; // To avoid the inclusion of '\0'
	printf("Length of the string is: %d\n", len);
	return 0;
}
```

Output:
```
Input the string: Hello, World!
Length of the string is: 13
```

Program 3: Write a program to separate individual characters from a string.
Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
	printf("Input a string: ");
	char str[SIZE];
	scanf("%[^\n]%c", str);

	int i;
	printf("The characters of the string are:\n");
	for(i = 0; str[i] != '\0'; i++) {
		printf("%c ", str[i]);
	}
	printf("\n");

	return 0;
}
```

Output:
```
Input a string: Hello, World!
The characters of the string are:
H e l l o ,   W o r l d !
```

Program 4: Write a program in C to print individual characters of string in reverse order.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
        printf("Input the string: ");
        char str[SIZE];
        scanf("%[^\n]%c", str);

        int i, len = 0;
        while(str[len++] != '\0');
        len--;

        for(i = len-1; i >= 0; i--) {
                printf("%c", str[i]);
        }
        printf("\n");

        return 0;
}
```

Output:

```
Input the string: Hello, World!
!dlroW ,olleH
```

Program 5: Write a program in C to count the total number of words in a string.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
        printf("Input the string: ");
        char str[SIZE];
        scanf("%[^\n]%c", str);

        int n = 0, i, count = 1;

        while(str[n++] != '\0');
        n -= 1;

        for(i = 0; i < n-1; i++) {
                if(str[i] == ' ' &&
                        ((str[i+1] >= 'A' && str[i] <= 'Z') ||
                         (str[i+1] >= 'a' && str[i]) <= 'z')) {
                        count++;
                }
        }
        printf("Total number of words in the string are: %d\n", count);

        return 0;
}
```

Output:

```
Input the string: A quick brown fox jumps over the lazy dog.
Total number of words in the string are: 9
```

Program 6: Write a program in C to compare two strings without using string library functions.

Code:
```
#include <stdio.h>
#define SIZE 100

int main() {
        char str1[SIZE], str2[SIZE];
        printf("Input the 1st string: ");
        fgets(str1, SIZE, stdin);
        printf("Input the 2nd string: ");
        fgets(str2, SIZE, stdin);
        int n1 = 0, n2 = 0, i;

        for(i = 0; str1[i] != '\0'; i++) {
                n1++;
        }
        for(i = 0; str2[i] != '\0'; i++) {
                n2++;
        }

        if(n1 == n2) {
                for(i = 0; i < n1; i++) {
                        if(str1[i] != str2[i]) {
                                printf("Strings are not equal.\n");
                                return -1;
                        }
                }
                printf("Strings are equal.\n");
        }
        else {
                printf("Strings are not equal.\n");
        }

        return 0;
}
```

Output:

```
Input the 1st string: Kunal Kumar Sahoo
Input the 2nd string: Kunal Kumar Sahoo
Strings are equal
```

Output:

```
Input the 1st string: Kunal Kumar Sahoo
Input the 2nd string: Kuna; Kumar Sahoo
Strings are not equal.
```

Program 7: Write a program in C to count total number of alphabets, digits and special characters in a string.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
        printf("Input the string: ");
        char str[SIZE];
        scanf("%[^\n]%c", str);

        int i = 0, alphabets = 0, digits = 0, spc_chars = 0;
        while(str[i++] != '\0') {
                if((str[i] >= 'A' && str[i] <= 'Z') ||
                        str[i] >= 'a' && str[i] <= 'z') {
                        alphabets++;
                }
                else if(str[i] >= '0' && str[i] <= '9') {
                        digits++;
                }
                else {
                        spc_chars++;
                }
        }
        printf("Number of Alphabets in the string is: %d\n", alphabets);
        printf("Number of Digits in the string is: %d\n", digits);
        printf("Number of Special characters in the string is: %d\n",
spc_chars);

        return 0;
}
```

Output:

```
Input the string: kunal.sce21@sot.pdpu.ac.in
Number of Alphabets in the string is: 18
Number of Digits in the string is: 2
Number of Special characters in the string is: 6
```

Program 8: Write a program in C to copy one string to another string.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
        printf("Input the string: ");
        char str[SIZE], str_copy[SIZE]; scanf("%[^\n]%c", str);

        int i = 0;
        while(str[i] != '\0') {
                str_copy[i] = str[i];
                i++;
        }

        printf("The First string is: %s\n", str);
        printf("The Second string is: %s\n", str_copy);
        printf("The numbers of characters copied: %d\n", i);

        return 0;
}
```

Output:

```
Input the string: I study in PDEU
The First string is: I study in PDEU
The Second string is: I study in PDEU
The numbers of characters copied: 15
```

Program 9: Write a program in C to count total number of vowel or consonant in a string.

Code:

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 100
int main() {
        printf("Input the string: ");
        char str[SIZE]; scanf("%[^\n]%c", str);
        int vowels = 0, consonants = 0;
        for(int i = 0; str[i] != '\0'; i++) {
           if(isalpha(str[i]))
               switch(tolower(str[i])) {
                  case 'a':
                  case 'e':
                  case 'i':
                  case 'o':
                  case 'u':
                     vowels++;
                     break;
                default:
                   consonants++;
         }
        }
        printf("The total number of vowels in the string is: %d\n", vowels);
        printf("The total number of consonants in the string is: %d\n",
consonants);

        return 0;
}
```

Output:

```
Input the string: a quick brown fox jumps over the lazy dog.
The total number of vowels in the string is: 11
The total number of consonants in the string is: 22
```

Program 10: Write a program in C to find maximum occurring character in a string.

Code:

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

#define SIZE 100

int main() {
	printf("Input the string: ");
	char str[SIZE];
	scanf("%[^\n]%c", str);

	int i, j, freq, max_freq;
	char mode_char;
	max_freq = 0;

	for(i = 0; str[i] != '\0'; i++) {
		freq = 0;
		for(j = 0; str[j] != '\0'; j++) {
			if(tolower(str[i]) == tolower(str[j])) {
				freq++;
			}
		}
		if(freq > max_freq) {
			mode_char = str[i];
			max_freq = freq;
		}
	}
```

```
        printf("The Highest frequency of character '%c'\nappears number of
times: %d\n", mode_char, max_freq);

        return 0;
}
```

Output:

```
Input the string: I can write programs in C language developed by Dennis Ritchie.
The Highest frequency of character ' '
appears number of times: 10
```

Output:

```
Input the string: Kunal Kumar Sahoo
The Highest frequency of character 'a'
appears number of times: 3
```

Program 11: Write a C program to sort a string array in ascending order.

Code:

```
#include <stdio.h>
#define SIZE 100
int main() {
        printf("Input the string: ");
        char str[SIZE];
        fgets(str, SIZE, stdin);
        int i, j, n=0; char temp;
        while(str[n++] != '\0'); n -= 2; // Remove the null and an extra increment
after that.
        for(i = 0; i < n-1; i++) {
                for(j = i+1; j < n; j++) {
                        if(str[i] > str[j]) {
                                temp = str[i];
                                str[i] = str[j];
                                str[j] = temp;
                        }
                }
        }
        printf("After sorting the string appears like: %s\n", str);
        return 0;
}
```

Output:

```
Input the string: Hello, World!
After sorting the string appears like:  ,HWdellloor!
```

Program 12: Write a program in C to read a strings through keyboard and sort the words.

Code:

```
#include <stdio.h>
#include <string.h>
#define SIZE 100

int main() {
        printf("Input number of strings: ");
        int N; scanf("%d", &N);

        char words[N][SIZE], temp[SIZE];

        int i, j;
        printf("Input %d strings:\n", N);
        for(i = 0; i <= N; i++)
                fgets(words[i], sizeof(words), stdin);
        for(i = 1; i <= N; i++) {
                for(j = 0; j <= N-i; j++) {
                        if(strcmp(words[j], words[j+1]) > 0) {
                                strcpy(temp, words[j]);
                                strcpy(words[j], words[j+1]);
                                strcpy(words[j+1], temp);
                        }
                }
        }
```

```
	printf("The strings appear after sorting:\n");
	for(i = 0; i <= N; i++) {
		printf("%s\n", words[i]);
	}

	return 0;
}
```

Output:

```
Input number of strings: 3
Input 3 strings:
kunal
kumar
sahoo
The strings appear after sorting:

kumar
kunal
sahoo
```

Program 13: Write a program in C to extract a substring from a given string.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
	printf("Input the string: ");
	char str[SIZE];
	scanf("%[^\n]%c", str);
	int start, length, i;
	printf("Input the position to start extraction: ");
	scanf("%d", &start);
	printf("Input the length of substring: ");
	scanf("%d", &length);
	printf("The substring retrieve from the string is: \"");
	for(i = start-1; i < start+length-1; i++) {
		printf("%c", str[i]);
	}
	printf("\"\n");
	return 0;
}
```

Output:

```
Input the string: A quick brown fox jumps over the lazy dog.
Input the position to start extraction: 2
Input the length of substring: 19
The substring retrieve from the string is: " quick brown fox ju"
```

Program 14: Write a C program to check whether a given substring is present in the given string.

Code:

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

#define SIZE 100

int main() {
	char str[SIZE], substr[SIZE];
	int len_str = 0, len_substr = 0, i, j;
	bool found;
	printf("Input the string: ");
	fgets(str, SIZE, stdin);
	printf("Input the substring to be searched: ");
	fgets(substr, SIZE, stdin);
	while(str[len_str++] != '\0'); len_str -= 2;
	while(substr[len_substr++] != '\0'); len_substr -= 2;
	for(i = 0; i <= len_str-len_substr; i++) {
		for(j = i; j < i+len_substr-1; j++) {
			found = true;
```

```
                        if(str[j] != substr[j-i]) {
                                found = false;
                                break;
                        }
                }
                if(found) break;
        }
        if(found) {
                printf("The substring exists in the string.\n");
        }
        else {
                printf("The substring does not exist in the string.\n");
        }

        return 0;
}
```

```
Output:
Input the string: This is a test string.
Input the substring to be searched: test
The substring exists in the string.
```

```
Output:
Input the string: This is a test string.
Input the substring to be searched: Hello
The substring does not exist in the string.
```

Program 15: Write a program in C to read a sentence and replace lowercase characters by uppercase and vice-versa.

Code:

```
#include <stdio.h>
#define SIZE 100

int main() {
        printf("Input the string: ");
        char str[SIZE];
        scanf("%[^\n]%c", str);

        printf("The given sentence is: %s\n", str);
        int i;
        for(i = 0; str[i] != '\0'; i++) {
                if(str[i] >= 'A' && str[i] <= 'Z') {
                        str[i] += ('a'-'A');
                }
                else if(str[i] >= 'a' && str[i] <= 'z') {
                        str[i] -= ('a'-'A');
                }
        }
        printf("After case swapping the string is: %s\n", str);

        return 0;
}
```

Output:

```
Input the string: Hello, World!
The given sentence is: Hello, World!
After case swapping the string is: hELLO, wORLD!
```

Program 16: Write a program in C to find the number of times a given word 'the' appears in the given string.

Code:
```c
#include <stdio.h>
#include <stdbool.h>

#define SIZE 100

int main() {
   int length = 0, i, frequency = 0;
   bool t, h, e, space;
   char str[SIZE];
   printf("Enter a string: ");
   fgets(str, SIZE, stdin);
   while(str[length++] != '\0'); length -= 2;

   for (i = 0; i < length-3; i++) {
      t =(str[i] == 't' || str[i] == 'T');
      h =(str[i + 1] == 'h' || str[i + 1] == 'H');
      e =(str[i + 2] == 'e'|| str[i + 2] == 'E');
      space =(str[i + 3] == ' ' || str[i + 3] == '\0');

      if(t && h && e && space) {
         frequency++;
      }
   }
   printf("Frequency of the word 'the' is %d\n", frequency);

   return 0;
}
```
Output:
```
Enter a string: The quick brown fox jumps over the lazy dog.
Frequency of the word 'the' is 2
```

Program 17: Write a program in C to remove characters in String Except Alphabets.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
	char str1[SIZE], str2[SIZE];
	printf("Input the string: ");
	scanf("%[^\n]%c", str1);

	int i = 0, j = 0;
	while(str1[i] != '\0') {
		if((str1[i] >= 'A' && str1[i] <= 'Z') ||
			(str1[i] >= 'a' && str1[i] <= 'z')) {
			str2[j++] = str1[i];
		}
		i++;
	}

	printf("After removing characters except alphabets, the output string
is: %s\n", str2);

	return 0;
}
```

Output:

```
Input the string: Hello, World!
After removing characters except alphabets, the output string is: HelloWorld
```

Program 18: Write a program in C to Find the Frequency of Characters.

Code:

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

#define SIZE 100

int main() {
        char str[SIZE], target;

        printf("Input the string: ");
        fgets(str, SIZE, stdin);
        printf("Input the character to find frequency: ");
        scanf("%c", &target);

        int i, freq = 0;

        for(i = 0; str[i] != '\0'; i++) {
                if(str[i] == toupper(target) ||
                        str[i] == tolower(target)) {
                        freq++;
                }
        }

        printf("The frequency of '%c' is: %d\n", target, freq);

        return 0;
}
```

Output:

```
Input the string: A quick brown fox jumps over the lazy dog.
Input the character to find frequency: o
The frequency of 'o' is: 4
```

Program 19: Write a program in C to Concatenate Two Strings Manually.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
        char str1[SIZE], str2[SIZE];
        printf("Input the first string: ");
        fgets(str1, SIZE, stdin);
        printf("Input the second string: ");
        fgets(str2, SIZE, stdin);

        int n1 = 0, n2 = 0, i;

        while(str1[n1++] != '\0'); n1 -= 2;
        while(str2[n2++] != '\0'); n2 -= 2;

        str1[n1] = ' ';

        for(i = 0; i < n2; i++) {
                str1[n1+1+i] = str2[i];
        }

        printf("After concatenation the string is:\n%s\n", str1);

        return 0;
}
```

Output:

```
Input the first string: Kunal Kumar
Input the second string: Sahoo
After concatenation the string is:
Kunal Kumar Sahoo
```

Program 20: Write a program in C to find the largest and smallest word in a string.

Code:
```
#include <stdio.h>
#include <string.h>
#define SIZE 100
int main() {
        char string[SIZE];
        printf("Input the string: ");
        fgets(string, SIZE, stdin);

        char words[SIZE][SIZE], small[SIZE], large[SIZE];
        int i = 0, j = 0, k;

        for(k = 0; string[k] != '\0'; k++) {
                if(string[k] != ' ' && string[k] != '\0') {
                        words[i][j++] = string[k];
                }
                else {
                        words[i][j] = '\0';
                        i++;
                        j = 0;
                }
        }
        strcpy(small, words[0]);
        strcpy(large, words[0]);
        for(k = 0; k <= i; k++) {
                if(strlen(small) > strlen(words[k]))
                        strcpy(small, words[k]);
                if(strlen(large) < strlen(words[k]))
                        strcpy(large, words[k]);
        }
        printf("The largest word is '%s'\nand the smallest word is '%s'\nin the
string: %s", large, small, string);
        return 0;
}
```

Output:
```
Input the string: This is a string with smallest and largest words.
The largest word is 'smallest'
and the smallest word is 'a'
in the string: This is a string with smallest and largest words.
```

Program 21: Write a program in C to convert a string to uppercase.

Code:
```c
#include <stdio.h>

#define SIZE 100

int main() {
   char string[SIZE];
   printf("Input a string in lowercase: ");
   scanf("%[^\n]%c", string);

   int i = 0;
   while(string[i] != '\0') {
      if(string[i] >= 'a' && string[i] <= 'z') {
         string[i] += ('A' - 'a');
      }
      i++;
   }
   printf("Here is the above string in UPPERCASE:\n%s\n", string);

   return 0;
}
```

Output:
```
Input a string in lowercase: Kunal Kumar Sahoo
Here is the above string in UPPERCASE:
KUNAL KUMAR SAHOO
```

Program 22: Write a program in C to convert a string to lowercase.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
   char string[SIZE];
   printf("Input a string in UPPERCASE: ");
   scanf("%[^\n]%c", string);

   int i = 0;
   while(string[i] != '\0') {
      if(string[i] >= 'A' && string[i] <= 'Z') {
         string[i] += ('a' - 'A');
      }
      i++;
   }
   printf("Here is the above string in lowercase:\n%s\n", string);

   return 0;
}
```

Output:

```
Input a string in UPPERCASE: KUNAL KUMAR SAHOO
Here is the above string in lowercase:
kunal kumar sahoo
```

Program 23: Write a program in C to check whether a letter is uppercase or not.

Code:

```
#include <stdio.h>

int main() {
        printf("Input a character: ");
        char c; scanf("%c", &c);

        if(c >= 'A' && c <= 'Z') {
                printf("The entered letter is an UPPERCASE letter.\n");
        }
        else {
                printf("The entered letter is not an UPPERCASE letter.\n");
        }

        return 0;
}
```

Output:

```
Input a character: K
The entered letter is an UPPERCASE letter.
```

Output:

```
Input a character: l
The entered letter is not an UPPERCASE letter.
```

Program 24: Write a program in C to replace the spaces of a string with a specific character.

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

#define SIZE 100

int main() {
	char str[SIZE], replacement_char;
	printf("Input a string: ");
	fgets(str, SIZE, stdin);
	printf("Input replacement character: ");
	scanf("%c", &replacement_char);

	int i = 0;
	while(str[i] != '\0') {
		if(str[i] == ' ') {
			str[i] = replacement_char;
		}
		i++;
	}

	printf("After replacing the space with %c the new string is:\n%s",
replacement_char, str);

	return 0;
}
```

Output:
```
Input a string: A quick brown fox jumps over the lazy dog.
Input replacement character: $
After replacing the space with $ the new string is:
A$quick$brown$fox$jumps$over$the$lazy$dog.
```

Program 25: Write a program in C to count the number of punctuation characters exists in a string.

Code:

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

#define SIZE 100

int main() {
        printf("Input the string: ");
        char str[SIZE]; scanf("%[^\n]%c", str);

        int count = 0, i;
        for(i = 0; str[i] != '\0'; i++) {
                if(ispunct(str[i])) {
                        count++;
                }
        }

        printf("The number of punctuation characters in the string is: %d\n",
count);

        return 0;
}
```

Output:

```
Input the string: A quick brown fox, jumps over, the lazy dog.
The number of punctuation characters in the string is: 3
```

Program 26: Write a program in C to print only the string before new line character.

Code:
```
#include <stdio.h>
#include <ctype.h>

int main() {
	int i = 0;
	char str[] = "The quick brown fox \n jumps over the \n lazy dog.";
	while(isprint(str[i])) {
		putchar(str[i++]);
	}
	printf("\n");

	return 0;
}
```

Output:
```
The quick brown fox
```

Program 27: Write a program in C to check whether a letter is lowercase or not.

Code:
```
#include <stdio.h>
int main() {
	printf("Input a character: ");
	char c; scanf("%c", &c);
	if(c >= 'a' && c <= 'z') {
		printf("The entered letter is a lowercase letter.\n");
	}
	else {
		printf("The entered letter is not a lowercase letter.\n");
	}
	return 0;
}
```

Output:

```
Input a character: k
The entered letter is a lowercase letter.
```

Output:

```
Input a character: K
The entered letter is not a lowercase letter.
```

Program 28: Write a program in C to check whether a character is digit or not.

Code:

```
#include <stdio.h>

int main() {
	printf("Input a character: ");
	char c; scanf("%c", &c);

	if(c >= '0' && c <= '9') {
		printf("The entered character is a digit.\n");
	}
	else {
		printf("The entered character is not a digit.\n");
	}

	return 0;
}
```

Output:

```
Input a character: 6
The entered character is a digit.
```

Output:

```
Input a character: @
The entered character is not a digit.
```

Program 29: Write a program in C to split string by space into words.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
        printf("Input a string: ");
        char str[SIZE]; fgets(str, SIZE, stdin);
        printf("Strings or words after split by spaces are:\n");
        int i = 0;
        while(str[i] != '\0') {
                if(str[i] == ' ') {
                        printf("\n");
                }
                else {
                        printf("%c", str[i]);
                }
                i++;
        }
        printf("\n");
        return 0;
}
```

Output:

```
Input a string: A quick brown fox jumps over the lazy dog.
Strings or words after split by spaces are:
A
quick
brown
fox
jumps
over
the
lazy
dog.
```

Program 30: Write a C programming to find the repeated character in a given string.

Code:
```
#include <stdio.h>
#define SIZE 100

int main() {
        char str[SIZE], temp, max_rep;
        printf("Input a string: ");
        scanf("%[^\n]%c", str);
        int count, max_count = 1, i, j;
        for(i = 0; str[i] != '\0'; i++) {
                temp = str[0];
                count = 1;
                for(j = i+1; str[j] != '\0'; j++) {
                        if(str[i] == str[j]) {
                                count++;
                        }
                }
                if(count > 1 && count > max_count) {
                        max_rep = str[i];
                        max_count = count;
                }
        }
        if(max_count > 1) {
                printf("The most repetitive character in '%s' is: %c\n", str,
max_rep);
        }
        else {
                printf("No character is repeated in '%s'\n", str);
        }
        return 0;
}
```
Output:
```
Input a string: Hello, World!
The most repetitive character in 'Hello, World!' is: l
```

Program 31: Write a C program to count of each character in a given string.

Code:
```
#include <stdio.h>
#include <stdbool.h>

#define SIZE 100

int main() {
	char str[SIZE], chars[SIZE];

	printf("Input a string: ");
	scanf("%[^\n]%c", str);
	int i, j, k, freq[SIZE];
	bool flag;
	chars[0] = str[0];
	freq[0] = 1;
	j = 0;
	for(i = 1; str[i] != '\0'; i++) {
		for(k = 0; k <= j; k++) {
			if(chars[k] == str[i]) {
				freq[k]++;
				flag = true;
				break;
			}
		}
		if(k > j) {
			flag = false;
		}
		if(!flag) {
			j++;
			chars[j] = str[i];
			freq[j] = 1;
		}

	}
	printf("The count of each character in the string %s is\n", str);
```

```
        for(i = 0; i <= j; i++) {
                printf("%c\t%d\n", chars[i], freq[i]);
        }

        return 0;
}
```

Code:

```
Input a string: Pandit Deendayal Energy University
The count of each character in the string Pandit Deendayal Energy University is
P	1
a	3
n	4
d	2
i	3
t	2
 	3
D	1
e	4
y	3
l	1
E	1
r	2
g	1
U	1
v	1
s	1
```

Program 32: Write a C programming to convert vowels into upper case character in a given string.

Code:

```
#include <stdio.h>

#define SIZE 100

int main() {
	printf("Input a string: ");
	char str[SIZE]; scanf("%[^\n]%c", str);

	int i = 0;
	while(str[i]) {
		switch(str[i]) {
			case 'a':
			case 'e':
			case 'i':
			case 'o':
			case 'u':
				str[i] += ('A'-'a');
		}
		i++;
	}

	printf("After converting vowels into upper case the sentence becomes:\n
%s\n", str);

	return 0;
}
```

Output:

```
Input a string: Pandit Deendayal Energy University.
After converting vowels into upper case the sentence becomes:
PAndIt DEEndAyAl EnErgy UnIvErsIty.
```

Aim: To study and implement concepts of Functions in C.

Program 1: Write a C program to find cube of any number using function.

Code:

```
#include <stdio.h>

int cube(int);

int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   printf("Cube of %d: %d\n", n, cube(n));

   return 0;
}

int cube(int a) {
   return a * a * a;
}
```

Output:

```
Enter a number: 11
Cube of 11: 1331
```

Program 2: Write a C program to find diameter, circumference and area of circle using functions.

Code:

```
#include <stdio.h>

const float PI = 3.14;

float diameter(float);
float circumference(float);
float area(float);

int main() {
   printf("Enter radius of the circle: ");
   float r; scanf("%f", &r);

   printf("Diameter of circle: %f\n", diameter(r));
   printf("Circumference of circle: %f\n", circumference(r));
   printf("Area of circle: %f\n", area(r));

   return 0;
}
float diameter(float radius) {
   return 2 * radius;
}
float circumference(float radius) {
   return PI * diameter(radius);
}
float area(float radius) {
   return PI * radius * radius;
}
```

Output:

```
Enter radius of the circle: 100
Diameter of circle: 200.000000
Circumference of circle: 628.000000
Area of circle: 31400.000000
```

Program 3: Write a C program to find maximum and minimum between two numbers using functions.

Code:
```
#include <stdio.h>
#include <stdlib.h>

int* max_min(int num1, int num2) {
   int* arr = (int*)malloc(2 * sizeof(int));
   if(num1 > num2) {
      arr[0] = num1;
      arr[1] = num2;
   }
   else {
      arr[0] = num2;
      arr[1] = num1;
   }

   return arr;
}

int main() {
   int a, b;
   printf("Enter two numbers: ");
   scanf("%d %d", &a, &b);
   int* ans = max_min(a, b);
   printf("Maximum number: %d\n", *ans);
   printf("Minimum number: %d\n", *(ans+1));

   free(ans);
   return 0;
}
```

Output:
```
Enter two numbers: 69 96
Maximum number: 96
Minimum number: 69
```

Program 4: Write a C program to check whether a number is even or odd using functions.

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

int checkOddEven(int n) {
   return n % 2;
}

void classifyOddEven(int n) {
   switch(checkOddEven(n)) {
      case 1:
         printf("%d is Odd\n", n);
         break;
      case 0:
         printf("%d is Even\n", n);
         break;
   }
}

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

   return 0;
}
```

Output:
```
Enter a number: 2436
2436 is Even
```

Output:
```
Enter a number: 123
123 is Odd
```

Program 5: Write a C program to check whether a number is prime, Armstrong or perfect number using functions.

Code:
```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

bool checkPrime(int n) {
   for(int i = 2; i <= sqrt(n); i++)
      if(n % i == 0)
         return false;

   return true;
}

bool checkArmstrong(int n) {
   int digits = log10(n)+1;
   int sum = 0;
   for(int temp = n; temp > 0; temp /= 10)
      sum += pow(temp % 10, digits);

   return n == sum;
}

bool checkPerfect(int n) {
   int sum = 0;
   for(int i = 1; i < n; i++)
      if(n % i == 0)
         sum += i;
   return sum == n;
}

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

```
    if(checkPrime(n))
        printf("%d is prime\n", n);
    else
        printf("%d is composite\n", n);
    if(checkArmstrong(n))
        printf("%d is an Armstrong number\n", n);
    else
        printf("%d is not an Armstrong number\n", n);
    if(checkPerfect(n))
        printf("%d is a Perfect number\n", n);
    else
        printf("%d is not a Perfect number\n", n);

    return 0;
}
```

```
Output:
Enter a number: 6
6 is composite
6 is an Armstrong number
6 is a Perfect number
```

```
Output:
Enter a number: 153
153 is composite
153 is an Armstrong number
153 is not a Perfect number
```

```
Output:
Enter a number: 7
7 is prime
7 is an Armstrong number
7 is not a Perfect number
```

Program 6: Write a C program to find all prime numbers between given interval using functions.

Code:
```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

bool checkPrime(int n) {
   for(int i = 2; i <= sqrt(n); i++)
      if(n % i == 0)
         return false;
   return true;
}

void printPrimes(int l, int u) {
   for(int i = l; i <= u; i++)
      if(checkPrime(i))
         printf("%d ", i);
   printf("\n");
}

int main() {
   printf("Enter lower limit: ");
   int a; scanf("%d", &a);

   printf("Enter upper limit: ");
   int b; scanf("%d", &b);

   printPrimes(a, b);

   return 0;
}
```

Output:
```
Enter lower limit: 10 100
Enter upper limit: 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Program 7: Write a C program to print all strong numbers between given interval using functions.

Code:
```
#include <stdio.h>
#include <stdbool.h>
int factorial(int n) {
   int product = 1;
   while(n >= 2) {
      product *= n--;
   }
   return product;
}
bool isStrongNumber(int n) {
   int sum = 0;
   for(int i = n; i > 0; i /= 10)
      sum += factorial(i % 10);
   return sum == n;
}
void printStrongNumbers(int lower, int upper) {
   for(int i = lower; i <= upper; i++)
      if(isStrongNumber(i))
         printf("%d ", i);
   printf("\n");
}
int main() {
   printf("Enter lower limit: ");
   int a; scanf("%d", &a);
   printf("Enter upper limit: ");
   int b; scanf("%d", &b);
   printStrongNumbers(a, b);
   return 0;
}
```
Output:
```
Enter lower limit: 1
Enter upper limit: 1000
1 2 145
```

Program 8: Write a C program to print all Armstrong numbers between given interval using functions.

Code:
```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

int countDigits(int n) {
   return log10(n) + 1;
}
bool isArmstrong(int n) {
   int sum = 0;
   int digits = countDigits(n);
   for(int temp = n; temp > 0; temp /= 10)
      sum += pow(temp % 10, digits);
   return sum == n;
}
void printArmstrong(int lower, int upper) {
   for(int i = lower; i <= upper; i++)
      if(isArmstrong(i))
         printf("%d ", i);
   printf("\n");
}
int main() {
   printf("Enter lower limit: ");
   int n1; scanf("%d", &n1);
   printf("Enter upper limit: ");
   int n2; scanf("%d", &n2);
   printArmstrong(n1, n2);
   return 0;
}
```

Output:
```
Enter lower limit: 1
Enter upper limit: 1000
1 2 3 4 5 6 7 8 9 153 370 371 407
```

Program 9: Write a C program to print all perfect numbers between given interval using functions.

Code:
```
#include <stdio.h>
#include <stdbool.h>

bool isPerfect(int n) {
   int sum = 1;
   for(int i = 2; i < n; i++)
      if(n % i == 0)
         sum += i;
   return sum == n;
}

void printPerfectNumbers(int start, int end) {
   for(int i = start; i <= end; i++)
      if(isPerfect(i))
         printf("%d ", i);
   printf("\n");
}

int main() {
   printf("Enter lower limit: ");
   int a; scanf("%d", &a);

   printf("Enter upper limit: ");
   int b; scanf("%d", &b);

   printPerfectNumbers(a, b);
   return 0;
}
```

Output:
```
Enter lower limit: 1
Enter upper limit: 100
1 6 28
```

Program 10: Write a C program to find power of any number using recursion.

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

long calcPower(int base, int power) {
   if(power == 0)
      return 1;
   return base * calcPower(base, power-1);
}

int main() {
   printf("Enter base number: ");
   int n; scanf("%d", &n);

   printf("Enter power: ");
   int x; scanf("%d", &x);

   printf("%d^%d = %ld\n", n, x, calcPower(n, x));

   return 0;
}
```

Output:
```
Enter base number: 2
Enter power: 10
2^10 = 1024
```

Program 11: Write a C program to print all natural numbers between 1 to n using recursion.

Code:

```c
#include <stdio.h>

void printNaturalNumbers(int n) {
   static int i = 1;
   if(i > n)
      return;
   printf("%d ", i++);
   printNaturalNumbers(n);
}

int main() {
   printf("Enter how many natural numbers you want to print: ");
   int n; scanf("%d", &n);

   printNaturalNumbers(n);
   printf("\n");

   return 0;
}
```

Output:

```
Enter how many natural numbers you want to print: 10
1 2 3 4 5 6 7 8 9 10
```

Program 12: Write a C program to print all even or odd numbers in given range using recursion.

Code:

```
#include <stdio.h>

void printOddEvenNumbers(int lower, int upper) {
   if(lower > upper) {
      printf("\n");
      return;
   }
   printf("%d ", lower);
   printOddEvenNumbers(lower+2, upper);
}

int main() {
   printf("Enter lower limit: ");
   int l; scanf("%d", &l);

   printf("Enter upper limit: ");
   int u; scanf("%d", &u);

   if(l % 2) {
      printf("Even numbers are:\n");
      printOddEvenNumbers(l+1, u);
      printf("Odd numbers are:\n");
      printOddEvenNumbers(l, u);
   }
   else {
      printf("Even numbers are:\n");
      printOddEvenNumbers(l, u);
      printf("Odd numbers are:\n");
      printOddEvenNumbers(l+1, u);
   }

   return 0;
}
```

Output:

```
Enter lower limit: 10
Enter upper limit: 50
Even numbers are:
10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
Odd numbers are:
11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
```

Program 13: Write a C program to find sum of all natural numbers between 1 to n using recursion.

Code:

```
#include <stdio.h>
long sum(int n) {
   if(n == 1)
      return 1;
   return n + sum(n-1);
}
int main() {
   printf("Enter number of natural numbers to be considered for sum: ");
   int n; scanf("%d", &n);
   printf("Sum of %d natural numbers: %ld\n", n, sum(n));
   return 0;
}
```

Output:

```
Enter number of natural numbers to be considered for sum: 100
Sum of 100 natural numbers: 5050
```

Program 14: Write a C program to find sum of all even or odd numbers in given range using recursion.

Code:

```
#include <stdio.h>
int sumOfEvenOdd(int start, int end) {
   if(start > end)
      return 0;
   return start + sumOfEvenOdd(start + 2, end);
}
int main() {
   printf("Enter lower limit: ");
   int l; scanf("%d", &l);
   printf("Enter upper limit: ");
   int u; scanf("%d", &u);
   if(l % 2) {
      printf("Sum of all even numbers between %d and %d: %d\n", l, u,
sumOfEvenOdd(l+1, u));
      printf("Sum of all odd numbers between %d and %d: %d\n", l, u,
sumOfEvenOdd(l, u));
   }
   else {
      printf("Sum of all even numbers between %d and %d: %d\n", l, u,
sumOfEvenOdd(l, u));
      printf("Sum of all odd numbers between %d and %d: %d\n", l, u,
sumOfEvenOdd(l+1, u));
   }
   return 0;
}
```

Output:

```
Enter lower limit: 10
Enter upper limit: 100
Sum of all even numbers between 10 and 100: 2530
Sum of all odd numbers between 10 and 100: 2475
```

Program 15: Write a C program to find reverse of any number using recursion.

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

int reverse(int num) {
   int digits = log10(num);
   if(num == 0)
      return 0;
   return (num % 10) * pow(10, digits) + reverse(num/10);
}

int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   printf("Reverse of %d: %d\n", n, reverse(n));

   return 0;
}
```

Output:
```
Enter a number: 123456
Reverse of 123456: 654321
```

Program 16: Write a C program to check whether a number is palindrome or not using recursion.

Code:
```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

int reverse(int num) {
   int digits = log10(num);
   if(num == 0)
      return 0;
   return (num % 10) * pow(10, digits) + reverse(num/10);
}

bool isPalindrome(int num) {
   return num == reverse(num);
}

int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   if(isPalindrome(n))
      printf("%d is a palindrome\n", n);
   else
      printf("%d is not a palindrome\n", n);

   return 0;
}
```

Output:
```
Enter a number: 1331
1331 is a palindrome
```

Output:
```
Enter a number: 123
123 is not a palindrome
```

Program 17: Write a C program to find sum of digits of a given number using recursion.

Code:

```
#include <stdio.h>

int sumOfDigits(int num) {
   if(num / 10 == 0)
      return num;
   return num % 10 + sumOfDigits(num / 10);
}

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

   printf("Sum of digits of %d: %d\n", n, sumOfDigits(n));

   return 0;
}
```

Output:

```
Enter a number: 12345
Sum of digits of 12345: 15
```

Program 18: Write a C program to find factorial of any number using recursion.

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

long factorial(int n) {
   if(n == 0 || n == 1)
      return 1;
   return n * factorial(n-1);
}

int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   if(n < 0)
      printf("Factorials of negative numbers don't exist\n");
   else
      printf("%d! = %ld\n", n, factorial(n));
   return 0;
}
```

Output:
```
Enter a number: 0
0! = 1
```

Output:
```
Enter a number: -1
Factorials of negative numbers don't exist
```

Output:
```
Enter a number: 5
5! = 120
```

Program 19: Write a C program to generate nth Fibonacci term using recursion.

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

typedef unsigned int uint;

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

int main() {
   printf("Enter which term of the series you want to print: ");
   uint n; scanf("%d", &n);
   printf("Fibonacci term at position %d: %d\n", n, fibonacci(n));

   return 0;
}
```

Output:
```
Enter which term of the series you want to print: 10
Fibonacci term at position 10: 34
```

Program 20: Write a C program to find GCD (HCF) of two numbers using recursion.

Code:

```
#include <stdio.h>

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

int main() {
   printf("Enter two numbers: ");
   int x, y; scanf("%d %d", &x, &y);
   printf("GCD of %d and %d: %d\n", x, y, gcd(x, y));
   return 0;
}
```

Output:

```
Enter two numbers: 45 23
GCD of 45 and 23: 1
```

Output:

```
Enter two numbers: 24 6
GCD of 24 and 6: 6
```

Program 21: Write a C program to find LCM of two numbers using recursion.

Code:
```c
#include <stdio.h>

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

int lcm(int a, int b) {
   return a / gcd(a, b) * b;
}

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

   printf("LCM of %d and %d: %d\n", x, y, lcm(x, y));

   return 0;
}
```

Output:
```
Enter two numbers: 23 46
LCM of 23 and 46: 46
```

Output:
```
Enter two numbers: 23 45
LCM of 23 and 45: 1035
```

Program 22: Write a C program to display all array elements using recursion.

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

void printArray(int *arr, int n) {
   static int i = 0;
   if(i == n) {
      printf("\n");
      return;
   }
   printf("%d ", *(arr+(i++)));
   printArray(arr, n);
}

int main() {
   printf("Input the size of the array: ");
   int n; scanf("%d", &n);

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

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

   return 0;
}
```

Output:
```
Input the size of the array: 10
Enter 10 elements for the array:
1 2 3 4 5 6 7 8 9 0
Given array is:
1 2 3 4 5 6 7 8 9 0
```

Program 23: Write a C program to find sum of elements of array using recursion.

Code:

```
#include <stdio.h>

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

int main() {
   printf("Enter the size of the array: ");
   int n; scanf("%d", &n);
   int arr[n];

   printf("Enter %d elements in the array:\n", n);
   for(int i = 0; i < n; i++)
      scanf("%d", arr+i);
   printf("Sum of elements of the array: %d\n", sumOfArrayElements(arr,
n));

   return 0;
}
```

Output:

```
Enter the size of the array: 10
Enter 10 elements in the array:
1 2 3 4 5 6 7 8 9 0
Sum of elements of the array: 45
```

Program 24: Write a C program to find maximum and minimum elements in array using recursion.

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

int min(int a, int b) {
   return a < b ? a : b;
}

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

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

int findMax(int arr[], int n) {
   if(n == 1)
      return arr[0];
   return max(arr[n-1], findMax(arr, n-1));
}
int main() {
   printf("Enter size of the array: ");
   int n; scanf("%d", &n);
   int arr[n];
   printf("Enter %d elements in the array:\n", n);
   for(int i = 0; i < n; i++)
      scanf("%d", arr+i);
   printf("Minimum element of the array: %d\n", findMin(arr, n));
   printf("Maximum element of the array: %d\n", findMax(arr, n));

   return 0;
}
```

Output:

```
Enter size of the array: 10
Enter 10 elements in the array:
10 9 7 8 4 5 6 2 3 1
Minimum element of the array: 1
Maximum element of the array: 10
```

Program 25: Write a program in C to show the simple structure of a function.

Code:

```c
#include <stdio.h>

int add(int, int);
int main() {
   printf("Enter two numbers: ");
   int a, b; scanf("%d %d", &a, &b);
   printf("%d + %d = %d\n", a, b, add(a, b));
   return 0;
}
int add(int num1, int num2) {
   return num1 + num2;
}
```

Output:

```
Enter two numbers: 25 69
25 + 69 = 94
```

Program 26: Write a program in C to find the square of any number using the function.

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

long square(int n) {
   return n * n;
}

int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   printf("%d ^ 2 = %ld\n", n, square(n));

   return 0;
}
```
Output:
```
Enter a number: 45
45 ^ 2 = 2025
```

Program 27: Write a program in C to swap two numbers using function.

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

void swap(int *a, int *b) {
   *a = *a + *b;
   *b = *a - *b;
   *a = *a - *b;
}

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

   printf("Original values:\n");
```

```
    printf("First number: %d\nSecond number: %d\n", a, b);

    swap(&a, &b);
    printf("After swapping values:\n");
    printf("First number: %d\nSecond number: %d\n", a, b);

    return 0;
}
```

Output:

```
Enter two numbers: 69 96
Original values:
First number: 69
Second number: 96
After swapping values:
First number: 96
Second number: 69
```

Program 28: Write a program in C to check a given number is even or odd using the function.

Code:

```
#include <stdio.h>

int isOdd(int n) {
    return n % 2;
}

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

    if(isOdd(n)) printf("%d is Odd\n", n);
    else printf("%d is Even\n", n);
    return 0;
}
```

Output:

```
Enter a number: 12
12 is Even
```

Output:

```
Enter a number: 53
53 is Odd
```

Program 29: Write a program in C to find the sum of the series 1!/1+2!/2+3!/3+4!/4+5!/5 using the function.

Code:

```c
#include <stdio.h>

long factorial(int n) {
   if(n == 0 || n == 1)
      return 1;
   return n * factorial(n-1);
}

long sum(int n) {
   if(n == 1)
      return 1;
   return factorial(n-1) + sum(n-1);
}

int main() {
   printf("Enter number of terms: ");
   int n; scanf("%d", &n);
   printf("Sum of first %d terms of the series: %ld\n", n, sum(n));

   return 0;
}
```

Output:

```
Enter number of terms: 5
Sum of first 5 terms of the series: 34
```

Program 30: Write a program in C to convert decimal number to binary number using the function.

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

long dec2bin(int dec) {
   if(dec == 0)
      return 0;
   return dec % 2 + 10 * dec2bin(dec / 2);
}

int main() {
   printf("Enter a number in decimal form: ");
   int n; scanf("%d", &n);
   printf("Binary form of %d: %ld\n", n, dec2bin(n));

   return 0;
}
```

Output:
```
Enter a number in decimal form: 123
Binary form of 123: 1111011
```

Program 31: Write a program in C to check whether a number is a prime number or not using the function.

Code:
```
#include <stdio.h>
#include <stdbool.h>

bool isPrime(int n) {
   for(int i = 2; i < n; i++)
      if(n % i == 0)
         return false;
   return true;
}
```

```
int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   if(isPrime(n))
      printf("%d is a Prime number\n", n);
   else
      printf("%d is a Composite number\n", n);

   return 0;
}
```

Code:

```
Enter a number: 53
53 is a Prime number
```

Code:

```
Enter a number: 21
21 is a Composite number
```

Program 32: Write a program in C to get the largest element of an array using the function.

Code:

```
#include <stdio.h>

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

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

int main() {
   printf("Enter the size of the array: ");
```

```
    int n; scanf("%d", &n); int arr[n];

    printf("Input %d elements in the array:\n", n);
    for(int i = 0; i < n; i++) {
        printf("element - %d: ", i);
        scanf("%d", arr+i);
    }
    printf("Largest element in the array: %d\n", findMax(arr, n));

    return 0;
}
```

Output:

```
Enter the size of the array: 10
Input 10 elements in the array:
element - 0: 2
element - 1: 5
element - 2: 3
element - 3: 4
element - 4: 8
element - 5: 7
element - 6: 5
element - 7: 10
element - 8: 1
element - 9: 9
Largest element in the array: 10
```

Program 33: Write a program in C to check armstrong and perfect numbers using the function.

Code:
```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

bool isArmstrong(int num) {
   int digits = log10(num) + 1;
   int sum = 0;
   for(int temp = num; temp > 0; temp /= 10)
      sum += (int)pow(temp % 10, digits);
   return sum == num;
}

bool isPerfect(int num) {
   int sum = 0;
   for(int i = 1; i < num; i++)
      if(num % i == 0)
         sum += i;
   return sum == num;
}

int main() {
   printf("Enter a number: ");
   int n; scanf("%d", &n);
   if(isArmstrong(n))
      printf("%d is an armstrong number\n", n);
   else
      printf("%d is not an armstrong number\n", n);
   if(isPerfect(n))
      printf("%d is a perfect number\n", n);
   else
      printf("%d is not a perfect number\n", n);
   return 0;
}
```

Output:

```
Enter a number: 6
6 is an armstrong number
6 is a perfect number
```

Output:

```
Enter a number: 153
153 is an armstrong number
153 is not a perfect number
```

Program 34: Write a program in C to print all perfect numbers in given range using the function.

Code:

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

bool isPerfect(int num) {
   int sum = 0;
   for(int i = 1; i < num; i++)
      if(num % i == 0)
         sum += i;
   return num == sum;
}

void printPerfectNumbers(int lower, int upper) {
   while(lower <= upper) {
      if(isPerfect(lower))
         printf("%d ", lower);
      lower++;
   }
   printf("\n");
}
```

```
int main() {
    printf("Enter lower limit: ");
    int l; scanf("%d", &l);
    printf("Enter upper limit: ");
    int u; scanf("%d", &u);
    printPerfectNumbers(l, u);
}
```

Output:

```
Enter lower limit: 1
Enter upper limit: 1000
6 28 496
```

Program 35: Write a program in C to check whether two given strings are an anagram.

Code:

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

#define MAX_SIZE 100

void swap(char *a, char *b) {
    char temp = *a;
    *a = *b;
    *b = temp;
}

void sort(char str[], int n) {
    for(int i = 0; i < n; i++)
        for(int j = 1; j < n-i; j++)
            if(str[j-1] > str[j])
                swap(str+j-1, str+j);
}
```

```
int compare(char str1[], char str2[]) {
   sort(str1, strlen(str1));
   sort(str2, strlen(str2));
   return strcmp(str1, str2);
}

int main() {
   char str1[MAX_SIZE], str2[MAX_SIZE], temp1[MAX_SIZE],
temp2[MAX_SIZE];
   printf("Input the first string: ");
   scanf("%s", str1);
   printf("Input the second string: ");
   scanf("%s", str2);

   strcpy(temp1, str1); strcpy(temp2, str2);

   if(compare(str1, str2) == 0)
      printf("%s and %s are anagrams\n", temp1, temp2);
   else
      printf("%s and %s are not anagrams\n", temp1, temp2);

   return 0;
}
```

Output:

```
Input the first string: pears
Input the second string: spear
pears and spear are anagrams
```

Output:

```
Input the first string: hello
Input the second string: worl
hello and worl are not anagrams
```

Program 36: Write a C programming to find out maximum and minimum of some values using function which will return an array.

Code:
```
#include <stdio.h>
#include <stdlib.h>
int *extremes(int arr[], int n) {
   int *ans = malloc(2 * sizeof(int));
   ans[0] = arr[0], ans[1] = arr[0];
   for(int i = 1; i < n; i++) {
      if(arr[i] > ans[0])
         ans[0] = arr[i];
      if(arr[i] < ans[1])
         ans[1] = arr[i];
   }
   return ans;
}
int main() {
   printf("How many numbers you want to compare? ");
   int n; scanf("%d", &n); int arr[n];
   printf("Input %d values:\n", n);
   for(int i = 0; i < n; i++)
      scanf("%d", arr+i);
   int *ans = extremes(arr, n);
   printf("Minimum value: %d\n", *(ans+1));
   printf("Maximum value: %d\n", *ans);
   free(ans);

   return 0;
}
```

Output:
```
How many numbers you want to compare? 4
Input 4 values:
12 10 16 12
Minimum value: 10
Maximum value: 16
```

Aim: To study and implement concepts of File Handling in C.

Program 1: Write a program in C to create and store information in a text file.

Code:

```
#include <stdio.h>

int main() {
   char text[100] = "A quick brown fox jumps over the lazy dog.";

   char path[] = "/home/kunalkumarsahoo/Programming/C-Programming/C-
Programming-Practicals/File-Handling/test1.txt";
   FILE *fileptr = fopen(path, "w");
   fprintf(fileptr, "%s", text);
   fclose(fileptr);
   printf("Content written to file successfully\n");

   return 0;
}
```

Output:

```
Content written to file successfully
```

Content of test1.txt:

```
A quick brown fox jumps over the lazy dog.
```

Program 2: Write a program in C to read an existing file

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

int main() {
   char path[] = "test1.txt";
   FILE *fptr = fopen(path, "r");
   char c;
   printf("Contents of the file are:\n");
   while((c = fgetc(fptr)) != EOF) {
      printf("%c", c);
   }
   fclose(fptr);
   printf("\n");

   return 0;
}
```

Output:
```
Contents of the file are:
A quick brown fox jumps over the lazy dog.
```

Program 3: Write a program in C to write multiple lines in a text file.

Code:

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

int main() {
   printf("Input the number of lines to be written: ");
   int i, n; scanf("%d", &n);
   char str[100], path[] = "test1.txt";
   FILE *fptr = fopen(path, "w");

   printf("Input %d lines:\n", n);
   for(i = 0; i <= n; i++) {
      fgets(str, 100, stdin);
      if(*str != '\n') {
         str[strcspn(str, "\n")] = 0;
         str[strlen(str)] = '\n';
         fputs(str, fptr);
      }
   }
   fclose(fptr);

   printf("%d lines written in the file successfully\n", n);

   return 0;
}
```

Output:

```
Input the number of lines to be written: 3
Input 3 lines:
Hello World
My name is Kunal
I am a C-programmer
3 lines written in the file successfully
```

Contents of test1.txt:

```
Hello World
My name is Kunal
I am a C-programmer
```

Program 4: Write a program in C to read the file and store the lines into an array

Code:

```
#include <stdio.h>
#include <string.h>
int main() {
   printf("Input the file to be opened: ");
   char path[100], c; scanf("%s", path);
   FILE *fptr = fopen(path, "r");
   char content[100][100];
   int i = 0, total = 0;
   while(fgets(content[i], 100, fptr)) {
      content[i][strlen(content[i])-1] = '\0';
      i++;
   }
   fclose(fptr);
   total = i;
   printf("The content of the file is:\n");
   for(i = 0; i < total; i++) {
      printf("%s\n", content[i]);
   }

   return 0;
}
```

Output:

```
Input the file to be opened: test1.txt
The content of the file is:
Hello World
My name is Kunal
I am a C-programmer
```

Program 5: Write a program in C to Find the Number of Lines in a Text File.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the file name to be opened: ");
   char path[100], temp[100]; scanf("%s", path);
   FILE *fptr = fopen(path, "r");

   int lines = 0;
   while(fgets(temp, 100, fptr)) {
      lines++;
   }
   fclose(fptr);

   printf("Number of lines in the file: %d\n", lines);

   return 0;
}
```

Output:

```
Input the file name to be opened: test1.txt
Number of lines in the file: 3
```

Contents of test1.txt:

```
Hello World
My name is Kunal
I am a C-programmer
```

Program 6: Write a program in C to find the content of the file and number of lines in a Text File.

Code:

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

int main() {
   printf("Input the file to be opened: ");
   char path[100], c; scanf("%s", path);
   FILE *fptr = fopen(path, "r");
   char content[100][100];

   int i = 0;
   while(fgets(content[i], 100, fptr)) {
      //content[i][strlen(content[i])-1] = '\0';
      printf("%s", content[i]);
      i++;
   }
   fclose(fptr);
   printf("Number of lines in the file: %d\n", i);

   return 0;
}
```

Output:

```
Input the file to be opened: test1.txt
Hello World
My name is Kunal
I am a C-programmer
Number of lines in the file: 3
```

Program 7: Write a program in C to count a number of words and characters in a file

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

int main() {
   printf("Input the file name to be opened: ");
   char path[100], c; scanf("%s", path);

   FILE *fptr = fopen(path, "r");
   int chars = 0, words = 0;
   while((c = fgetc(fptr)) != EOF) {
      if(c == ' ' || c == '\n') words++;
      else chars++;
   }
   words++; //The word just before EOF gets missed.
   fclose(fptr);

   printf("Number of words: %d\n", words);
   printf("Number of characters: %d\n", chars);

   return 0;
}
```

Output:
```
Input the file name to be opened: test1.txt
Number of words: 11
Number of characters: 39
```

Contents of test1.txt:
```
Hello World
My name is Kunal
I am a C-programmer
```

Program 8: Write a program in C to delete a specific line from a file.

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

int main() {
   printf("Input the file name to be opened: ");
   char path[100]; scanf("%s", path);
   printf("Input the line number you want to remove: ");
   int n; scanf("%d", &n);

   FILE *fptr = fopen(path, "r");
   char content[100][100];
   int i = 0, total = 0;
   while(fgets(content[i], 100, fptr)) {
      content[i][strlen(content[i])-1] = '\0';
      i++;
   }
   total = i;
   fclose(fptr);

   fptr = fopen(path, "w");
   for(int i = 0; i < total; i++) {
      if(i != n-1) {
         fprintf(fptr, "%s\n", content[i]);
      }
   }
   fclose(fptr);

   printf("Line deleted successfully\n");

   return 0;
}
```

Output:

```
Input the file name to be opened: test1.txt
Input the line number you want to remove: 1
Line deleted successfully
```

Contents of test.txt:

```
My name is Kunal
I am a C-programmer
```

Program 9: Write a program in C to replace a specific line with another text in a file

Code:

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

int main() {
   printf("Input the file name to be opened: ");
   char path[100]; fgets(path, 100, stdin); path[strlen(path)-1] = '\0';
   printf("Input the content of the new line: ");
   char text[100]; fgets(text, 100, stdin); text[strlen(text)-1] = '\0';
   printf("Input the line number you want to replace: ");
   int n; scanf("%d", &n);
   FILE *fptr = fopen(path, "r");
   char content[100][100];
   int i = 0, total;
   while(fgets(content[i], 100, fptr)) {
      content[i][strlen(content[i])-1] = '\0';
      i++;
   }
   total = i;
   fclose(fptr);
   fptr= fopen(path, "w");
   for(i = 0; i <= total; i++) {
      if(i == n-1) {
         fprintf(fptr, "%s\n", text);
      }
```

```
        else {
            fprintf(fptr, "%s\n", content[i]);
        }
    }
    fclose(fptr);

    printf("Line replaced successfully\n");

    return 0;
}
```

Output:

```
Input the file name to be opened: test1.txt
Input the content of the new line: I am learning C
Input the line number you want to replace: 2
Line replaced successfully

Contents of test1.txt:
My name is Kunal
I am learning C
```

Program 10: Write a program in C to append multiple lines at the end of a text file.

Code:

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

int main() {
    printf("Input the file name to be opened: ");
    char path[100]; fgets(path, 100, stdin); path[strlen(path)-1] = '\0';
    printf("Input the number of lines to be written: ");
    int n, i; scanf("%d", &n);

    char text[n][100];
    printf("Input %d lines:\n", n);
    for(i = 0; i <= n; i++) {
        fgets(text[i], 100, stdin);
        text[i][strcspn(text[i], "\n")] = 0;
    }

    FILE *fptr = fopen(path, "a");
    for(i = 1; i <= n; i++) {
        fprintf(fptr, "%s\n", text[i]);
    }
    fclose(fptr);

    printf("Lines appended successfully\n");

    return 0;
}
```

Output:

```
Input the file name to be opened: test1.txt
Input the number of lines to be written: 2
Input 2 lines:
Hello, World!
I study at PDEU
Lines appended successfully

Contents of test1.txt
My name is Kunal
I am learning C

Hello, World!
I study at PDEU
```

Program 11: Write a program in C to copy a file in another name.

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

int main() {
   printf("Input source file name: ");
   char path1[100]; scanf("%s", path1);
   FILE *sourceptr = fopen(path1, "r");

   printf("Input target file name: ");
   char path2[100]; scanf("%s", path2);
   FILE *targetptr = fopen(path2, "w");

   char c;
   while((c = fgetc(sourceptr)) != EOF) {
      fputc(c, targetptr);
   }
   fclose(sourceptr); fclose(targetptr);
   printf("Content copied successfuly\n");

   return 0;
}
```

Output:
```
Input source file name: test1.txt
Input target file name: test2.txt
Content copied successfully
```

Contents of test1.txt:
```
My name is Kunal
I am learning C

Hello, World!
I study at PDEU
```

Contents of test2.txt:
My name is Kunal
I am learning C

Hello, World!
I study at PDEU

Program 12: Write a program in C to merge two files and write it in a new file.

Code:

```
#include <stdio.h>

int main() {
   printf("Input first file: ");
   char path1[100]; scanf("%s", path1);
   FILE *fptr1 = fopen(path1, "r");
   printf("Input second file: ");
   char path2[100]; scanf("%s", path2);
   FILE *fptr2 = fopen(path2, "r");
   printf("Input the name for the merged file: ");
   char path3[100]; scanf("%s", path3);
   FILE *fptr3 = fopen(path3, "w");
   char c;
   while((c = fgetc(fptr1)) != EOF) {
      fputc(c, fptr3);
   }
   fputc('\n', fptr3);
   fclose(fptr1);
   while((c = fgetc(fptr2)) != EOF) {
      fputc(c, fptr3);
   }
   fclose(fptr2);
   printf("Succesfully merged two files into another file\n");
   return 0;
}
```

```
Output:
Input first file: test1.txt
Input second file: test2.txt
Input the name for the merged file: test3.txt
Succesfully merged two files into another file

Contents of test1.txt:
My name is Kunal
I am learning C

Hello, World!
I study at PDEU

Contents of test2.txt:
Languages I know:
Python3
C++
C
Java
HTML
Bash

Contents of test3.txt
My name is Kunal
I am learning C

Hello, World!
I study at PDEU

Languages I know:
Python3
C++
C
Java
HTML
Bash
```

Program 13: Write a program in C to decrypt a previously encrypted file.

Code:

```
#include <stdio.h>

int main() {
   printf("Input the name of the file to be encrypted: ");
   char path[100]; scanf("%s", path);

   FILE *fptr = fopen(path, "r");
   long length, i;
   fseek(fptr, 0, SEEK_END);
   length = ftell(fptr);
   fseek(fptr, 0, SEEK_SET);

   char text[length+1];
   fread(text, 1, length, fptr);
   text[length] = '\0';
   fclose(fptr);

   fptr = fopen(path, "w");
   for(i = 0; text[i] != '\0'; i++) {
      fputc(text[i] - 69, fptr);
   }
   fclose(fptr);
   printf("File decrypted successfully\n");

   return 0;
}
```

Output:

```
Input the name of the file to be encrypted: test2.txt
File decrypted successfully
```

```
Contents of test2.txt before encryption:
Languages I know:
Python3
C++
C
Java
HTML
Bash

Contents of test2.txt after encryption:
##)"0#" .Û#Û&)*2õÅ#4/#*)îÅþææÅþÅ##1#Å####Åý#.#Å
```

Program 14: Write a program in C to decrypt a previously encrypted file.

Code:

```c
#include <stdio.h>
int main() {
   printf("Input the name of the file to be encrypted: ");
   char path[100]; scanf("%s", path);
   FILE *fptr = fopen(path, "r");
   long length, i;
   fseek(fptr, 0, SEEK_END);
   length = ftell(fptr);
   fseek(fptr, 0, SEEK_SET);
   char text[length+1];
   fread(text, 1, length, fptr);
   text[length] = '\0';
   fclose(fptr);
   fptr = fopen(path, "w");
   for(i = 0; text[i] != '\0'; i++) {
      fputc(text[i] - 69, fptr);
   }
   fclose(fptr);
   printf("File decrypted successfully\n");

   return 0;
}
```

Output:

```
Input the name of the file to be encrypted: test2.txt
File decrypted successfully

Contents of test2.txt before decryption:
##)"0#" .Û#Û&)*2õÅ#4/#*)îÅþææÅþÅ##1#Å####Åý#.#Å

Contents of test2.txt after decryption:
Languages I know:
Python3
C++
C
Java
HTML
Bash
```

Program 15: Write a program in C to remove a file from the disk

Code:

```
#include<stdio.h>

int main() {
   printf("Input the name of the file to be deleted: ");
   char path[100]; scanf("%s", path);

   int status = remove(path);

   if(status == 0){
      printf("%s file deleted successfully.\n", path);
   }
   else
   {
      printf("Unable to delete the file\n");
   }

   return 0;
}
```

Output:

```
Input the name of the file to be deleted: test3.txt
test3.txt file deleted successfully.
```

Text files before execution of program:

```
$ ls | grep .txt
test1.txt
test2.txt
test3.txt
```

Text files after execution of program:

```
$ ls | grep .txt
test1.txt
test2.txt
```