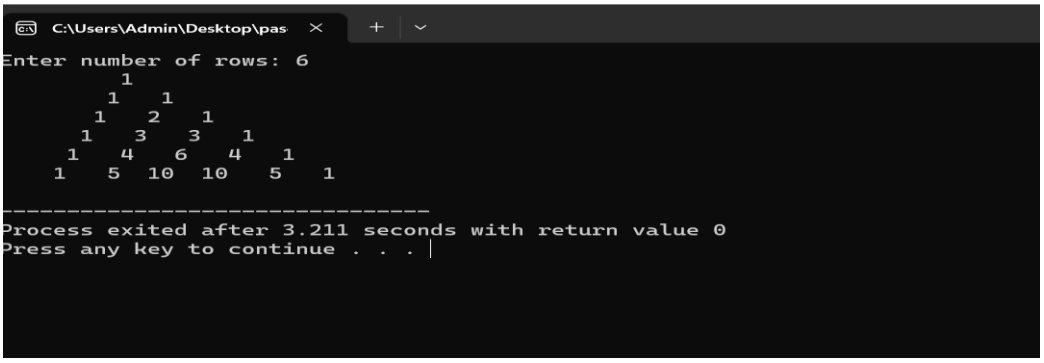
1.Write c program for pascal triangle?

Input:

```
#include <stdio.h>
int main() {
   int rows, coef = 1;
   printf("Enter number of rows: ");
   scanf("%d", &rows);
   for (int i = 0; i < rows; i++) {
      for (int space = 1; space <= rows - i; space++) {
         printf(" ");
      }
      for (int j = 0; j <= i; j++) {
         if (j == 0 || i == 0) {
            coef = 1;
         } else {
            coef = coef * (i - j + 1) / j;
         }
         printf("%4d", coef);
      }
      printf("\n");
   }
   return 0;
}
```

Output:



2.WRITE C PROGRAM FOR RIGHT TRIANGLE?

INPUT:

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter n: ");
   scanf("%d", &n);
   for (int i = 1; i <= n; i++) {
      for (int space = 1; space <= n - i; space++) {
         printf(" ");
      }
      for (int j = 1; j <= i; j++) {
         printf("*");
      }
      printf("\n");
   }
   return 0;
}
```

OUTPUT:

```
C:\Users\Admin\Desktop\RIG
Enter n: 5
    *
   **
  ***
 ****
*****

--------------------------------
Process exited after 2.405 seconds with return value 0
Press any key to continue . . .
```

3.WRITE C PROGRAM FOR THE FLOYDS TRIANGLE?

INPUT:

```
#include <stdio.h>


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

OUTPUT:

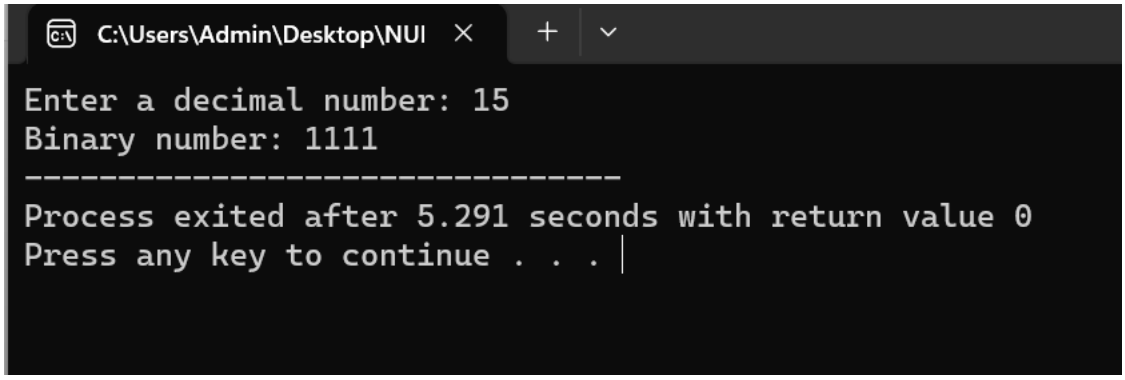
```
C:\Users\Admin\Desktop\FLO

Enter n: 5
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15

--------------------------------
Process exited after 1.889 seconds with return value 0
Press any key to continue . . .
```

4.WRITE C PROGRAM FOR TO CONVERT NUMBER TO BINARY?

INPUT:

```
#include <stdio.h>


int main() {
   int n, binary = 0, i = 1;
   printf("Enter a decimal number: ");
   scanf("%d", &n);
   while (n > 0) {
      binary += (n % 2) * i;
      n /= 2;
      i *= 10;
   }
   printf("Binary number: %d", binary);
   return 0;
}
```

OUTPUT:

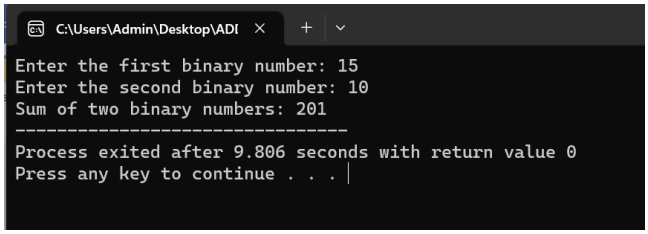


5. WRITE C PROGRAM TO ADD TWO BINARY NUMBERS?

INPUT:

```
#include <stdio.h>
int main() {
   long int binary1, binary2;
   int i = 0, remainder = 0, sum[20];
   printf("Enter the first binary number: ");
   scanf("%ld", &binary1);
   printf("Enter the second binary number: ");
   scanf("%ld", &binary2);
   while (binary1 != 0 || binary2 != 0) {
      sum[i++] = (binary1 % 10 + binary2 % 10 + remainder) % 2;
      remainder = (binary1 % 10 + binary2 % 10 + remainder) / 2;
      binary1 /= 10;
      binary2 /= 10;
   }
   if (remainder != 0) {
      sum[i++] = remainder;
   }
   i--;
   printf("Sum of two binary numbers: ");
   while (i >= 0) {
      printf("%d", sum[i--]);
   }
   return 0;
}
```

OUTPUT:

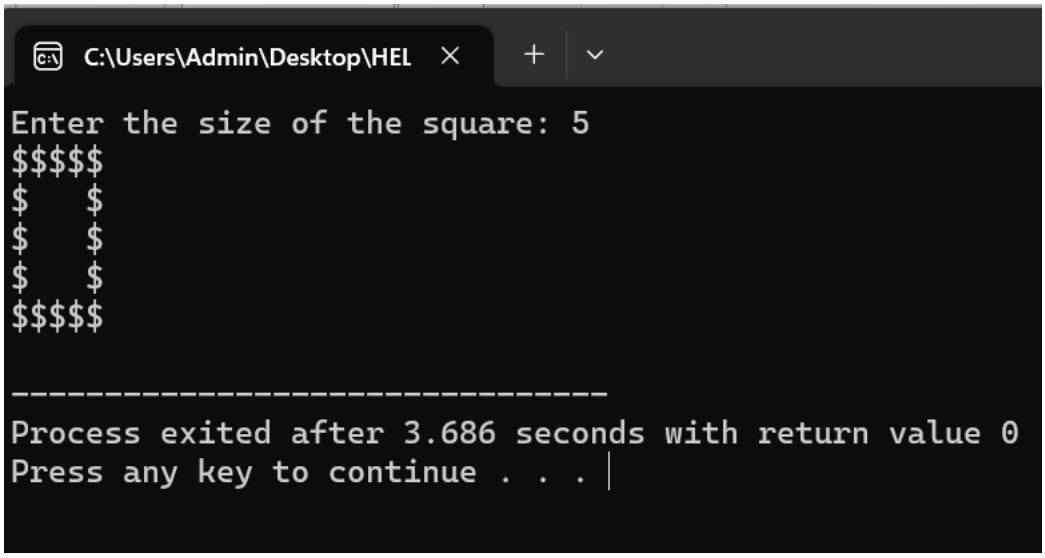


6.WRITE C PROGRAM TO HELLOW SQUARE USING DOLLAR?

INTPUT:

```
#include <stdio.h>


int main() {
   int n, i, j;
   printf("Enter the size of the square: ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++) {
      for (j = 1; j <= n; j++) {
         if (i == 1 || i == n || j == 1 || j == n) {
            printf("$");
         } else {
            printf(" ");
         }
      }
      printf("\n");
   }
   return 0;
}
```

OUTPUT:

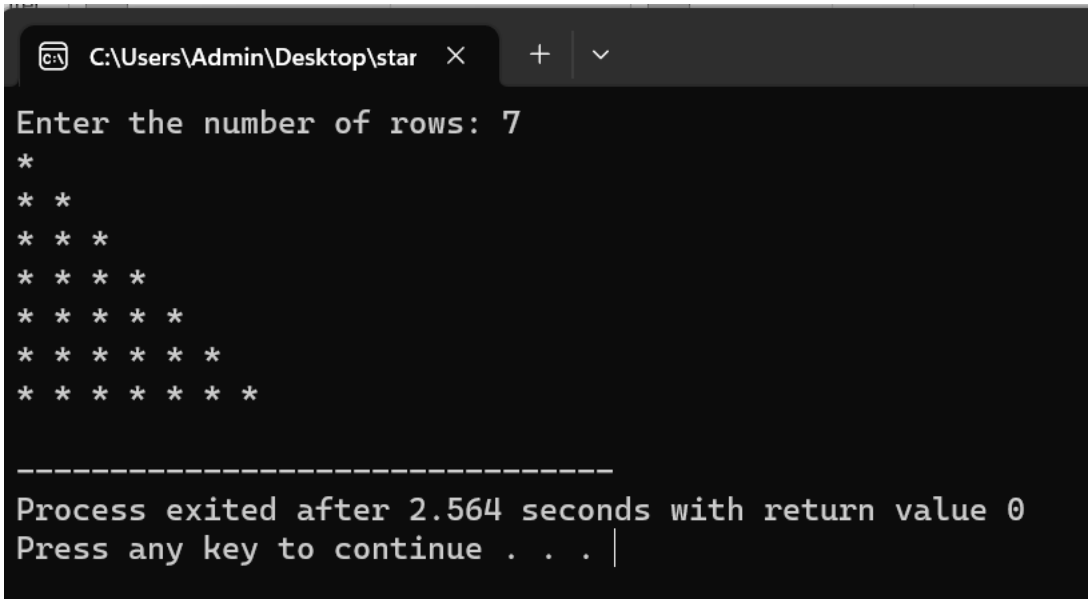


7.WRITE C PROGRAM TO PRINT STAR PATTERN?

INPUT:

```
#include <stdio.h>


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

OUTPUT:



8.rhombus in star pattern?

Input:

```
#include <stdio.h>
int main() {
   int n, i, j, space;
   printf("Enter the number of rows: ");
   scanf("%d", &n);
   space = n - 1;
   for (i = 0; i < n; i++) {
      for (j = 0; j < space; j++) {
         printf(" ");
      }
      for (j = 0; j <= i; j++) {
         printf("* ");
      }
      printf("\n");
      space--;
   }
   space = 0;
   for (i = n; i > 0; i--) {
      for (j = 0; j < space; j++) {
         printf(" ");
      }
      for (j = 0; j < i; j++) {
         printf("* ");
      }
      printf("\n");
      space++;
   }
   return 0;
}
```

Output:

```
C:\Users\Admin\Desktop\rho
Enter the number of rows: 5
    *
   * *
  * * *
 * * * *
* * * * *
* * * * *
 * * * *
  * * *
   * *
    *
```

9. WRITE C PROGRAM FOR NUMBERS STAR PATTERN?

INPUT:

```
#include <stdio.h>


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

OUTPUT:

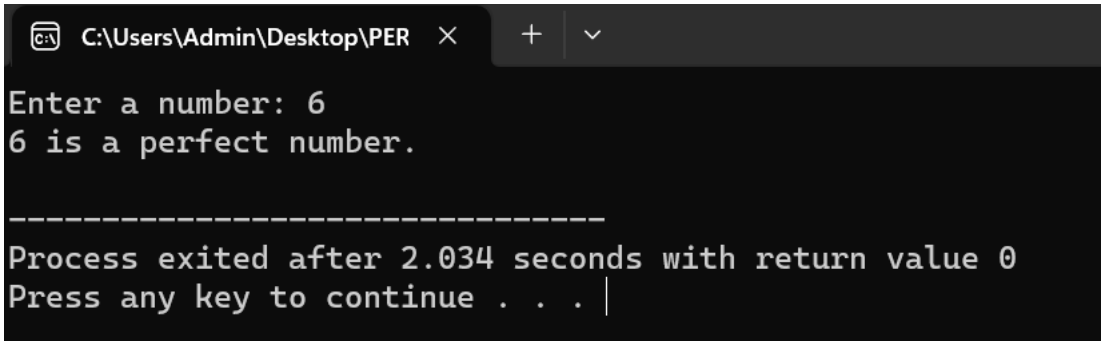
```
C:\Users\Admin\Desktop\nun
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

--------------------------------
Process exited after 0.02192 seconds with return value 0
Press any key to continue . . .
```

10.WRITE C PROGRAM TO FIND PERFECT NUMBER?

INPUT:

```
#include <stdio.h>


int main() {
   int n, i, sum = 0;
   printf("Enter a number: ");
   scanf("%d", &n);
   for (i = 1; i <= n / 2; i++) {
      if (n % i == 0) {
         sum += i;
      }
   }
   if (sum == n) {
      printf("%d is a perfect number.\n", n);
   } else {
      printf("%d is not a perfect number.\n", n);
   }
   return 0;
}
```

OUTPUT:

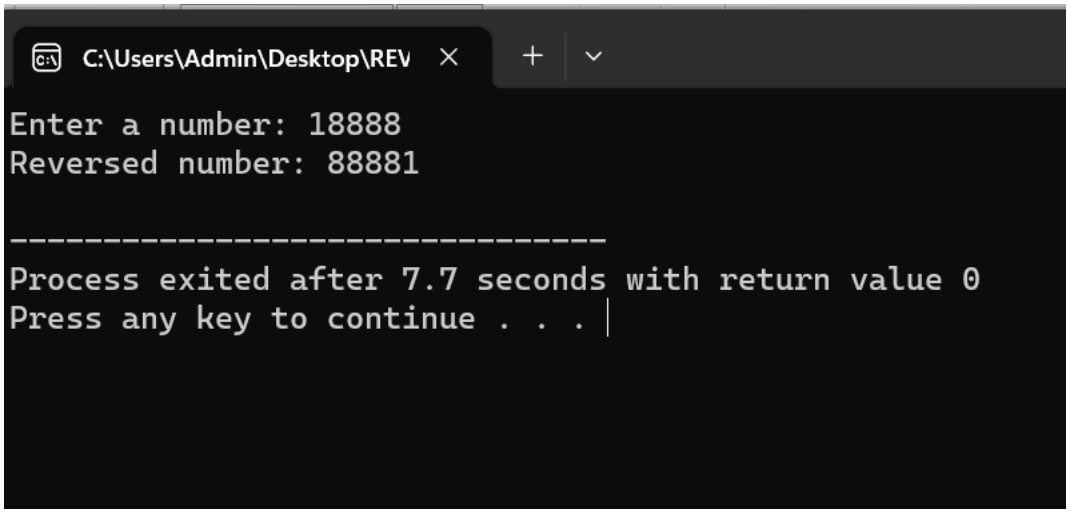


11. WRITE C PROGRAM TO CHECK REVERSE NUMBER?

INPUT:

```
#include <stdio.h>


int main() {
   int n, reversed = 0;
   printf("Enter a number: ");
   scanf("%d", &n);
   while (n != 0) {
      reversed = reversed * 10 + n % 10;
      n /= 10;
   }
   printf("Reversed number: %d\n", reversed);
   return 0;
}
```

OUTPUT:

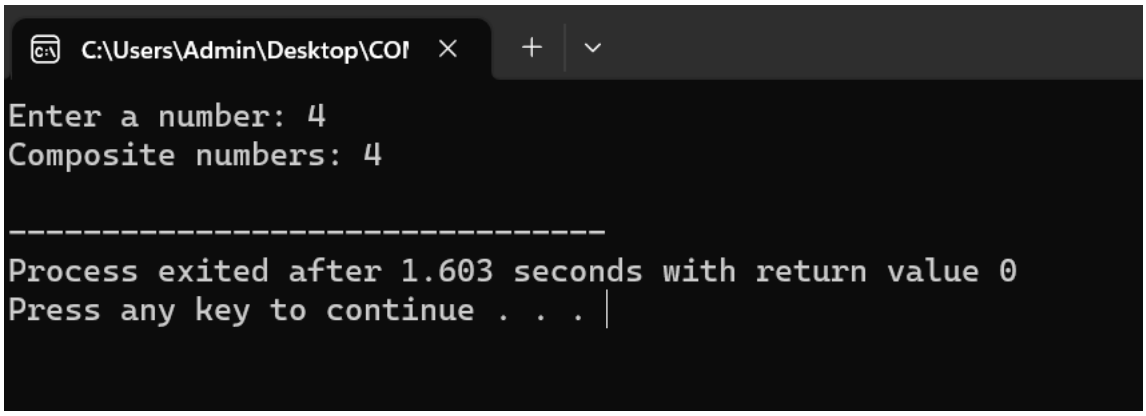


12. WRITE C PROGRAM FOR TO FIND COMPOSITE NUMBER?

INPUT:

```
#include <stdio.h>
int main() {
   int n, i, j, count;
   printf("Enter a number: ");
   scanf("%d", &n);
   printf("Composite numbers: ");
   for (i = 2; i <= n; i++) {
      count = 0;
      for (j = 2; j <= i / 2; j++) {
         if (i % j == 0) {
            count++;
            break;
         }
      }
      if (count != 0) {
         printf("%d ", i);
      }
   }
   printf("\n");
   return 0;
}
```

OUTPUT:

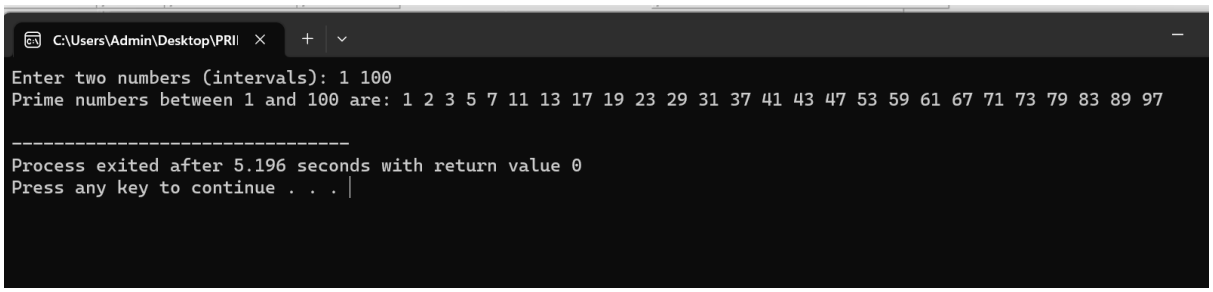


13. write c program to check the prime to given range?

Input:

```
#include <stdio.h>


int main() {
   int low, high, i, flag;
   printf("Enter two numbers (intervals): ");
   scanf("%d %d", &low, &high);
   printf("Prime numbers between %d and %d are: ", low, high);
   while (low < high) {
      flag = 0;
      for (i = 2; i <= low / 2; ++i) {
         if (low % i == 0) {
            flag = 1;
            break;
         }
      }
      if (flag == 0) {
         printf("%d ", low);
      }
      ++low;
   }
   printf("\n");
   return 0;
}
```

OUTPUT:

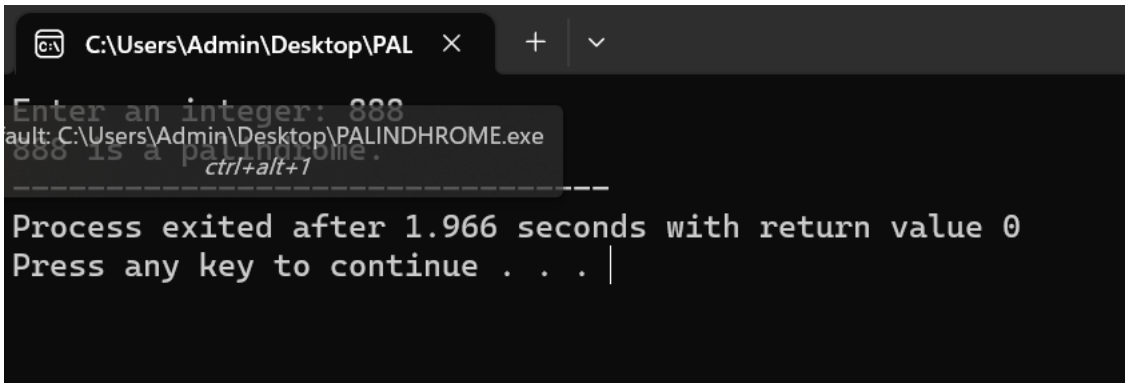


14.WRITE C PROGRAM FOR TO CHECK WHETHER IT IS A PALINDROME?

INPUT:

```
#include <stdio.h>


int main() {
   int n, reversedN = 0, remainder, originalN;
   printf("Enter an integer: ");
   scanf("%d", &n);
   originalN = n;
   while (n != 0) {
      remainder = n % 10;
      reversedN = reversedN * 10 + remainder;
      n /= 10;
   }
   if (originalN == reversedN) {
      printf("%d is a palindrome.", originalN);
   }
   else {
      printf("%d is not a palindrome.", originalN);
   }
   return 0;
}
```

OUTPUT:



15. WRITE C PROGRAM FOR LCM AND GCD?

INPUT:

```
#include <stdio.h>
int gcd(int a, int b) {
   while (b != 0) {
      int temp = b;
      b = a % b;
      a = temp;
   }
   return a;
}
int lcm(int a, int b) {
   return (a * b) / gcd(a, b);
}
int main() {
   int num1, num2;
   printf("Enter two positive integers: ");
   scanf("%d %d", &num1, &num2);
   if (num1 <= 0 || num2 <= 0) {
      printf("Both numbers should be positive integers.\n");
      return 1;
   }

   int result_gcd = gcd(num1, num2);
   int result_lcm = lcm(num1, num2);
   printf("GCD of %d and %d is: %d\n", num1, num2, result_gcd);
   printf("LCM of %d and %d is: %d\n", num1, num2, result_lcm);
   return 0;
}
```

OUTPUT:

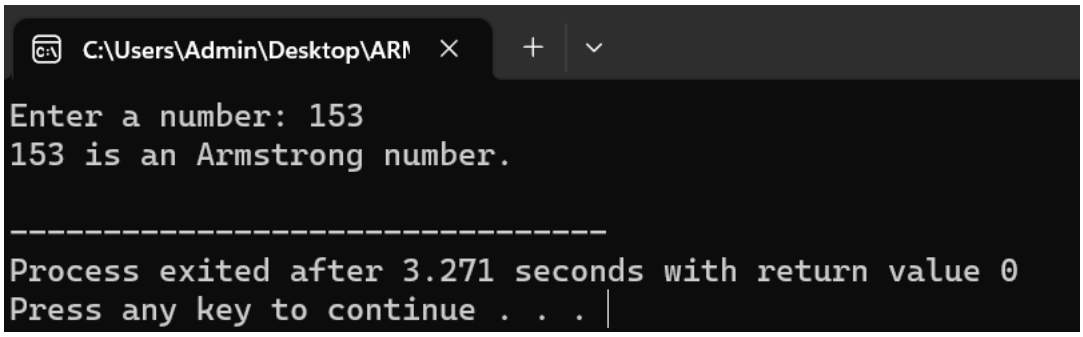
```mathematica
Enter two positive integers: 12 18
GCD of 12 and 18 is: 6
LCM of 12 and 18 is: 36
```

16. WRITE C PROGRAM FOR THE ARMSTRONG NUMBER?

```c
#include <stdio.h>
#include <math.h>
int countDigits(int num) {
    int count = 0;
    while (num != 0) {
        num /= 10;
        count++;
    }
    return count;
}
int isArmstrong(int num) {
    int originalNum = num;
    int numDigits = countDigits(num);
    int sum = 0;

    while (num != 0) {
        int digit = num % 10;
        sum += pow(digit, numDigits);
        num /= 10;
    }

    return (sum == originalNum);
}
```

```
int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);
    if (isArmstrong(number)) {
        printf("%d is an Armstrong number.\n", number);
    } else {
        printf("%d is not an Armstrong number.\n", number);
    }
    return 0;
}
```

OUTPUT:



17. WRITE C PROGRAM FOR FOR COSINE SERIES UPTO TERMS USING RECURSION?

INPUT:

```
#include <stdio.h>
#include <math.h>
int factorial(int num) {
   if (num == 0 || num == 1) {
      return 1;
   } else {
      return num * factorial(num - 1);
   }
}
double power(double base, int exponent) {
   if (exponent == 0) {
      return 1;
   } else if (exponent > 0) {
      return base * power(base, exponent - 1);
   } else {
      return 1 / power(base, -exponent);
   }
}
double cosineSeries(double x, int terms) {
   double sum = 0.0;
   int sign = 1;

   for (int i = 0; i < terms; i++) {
      double numerator = power(x, 2 * i);
      double denominator = factorial(2 * i);
      double term = sign * (numerator / denominator);
      sum += term;
      sign = -sign;
   }
```

```
    return sum;
}

int main() {
    double x;
    int terms;

    printf("Enter the value of x in radians: ");
    scanf("%lf", &x);

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

    double result = cosineSeries(x, terms);

    printf("Cosine of %lf radians using %d terms is: %lf\n", x, terms, result);

    return 0;
}
```

OUTPUT:

```
C:\Users\Admin\Desktop\COS

Enter the value of x in radians: 5
Enter the number of terms in the series: 5
Cosine of 5.000000 radians using 5 terms is: 2.528398

--------------------------------
Process exited after 9.404 seconds with return value 0
Press any key to continue . . .
```

18. Check Whether a Number is Positive or Negative?

INPUT:

```
#include <stdio.h>

int main() {
   int number;

   printf("Enter a number: ");
   scanf("%d", &number);

   if (number > 0) {
      printf("%d is a positive number.\n", number);
   } else if (number < 0) {
      printf("%d is a negative number.\n", number);
   } else {
      printf("The number is zero.\n");
   }

   return 0;
}
```

OUTPUT:

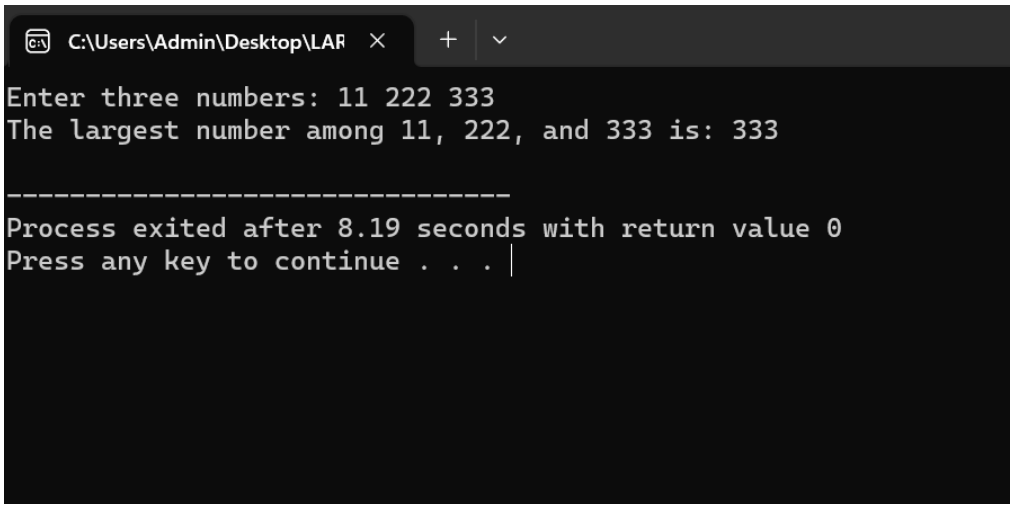
```
Enter a number: 5
5 is a positive number.

Enter a number: -8
-8 is a negative number.

Enter a number: 0
The number is zero.
```

19. Find the Largest Number Among Three Numbers.

INPUT:

```
#include <stdio.h>
int max(int a, int b) {
   return (a > b) ? a : b;
}
int findLargest(int a, int b, int c) {
   int largest = max(a, b);
   largest = max(largest, c);
   return largest;
}

int main() {
   int num1, num2, num3;

   printf("Enter three numbers: ");
   scanf("%d %d %d", &num1, &num2, &num3);

   int largestNumber = findLargest(num1, num2, num3);

   printf("The largest number among %d, %d, and %d is: %d\n", num1, num2,
num3, largestNumber);

   return 0;
}
```

OUTPUT:

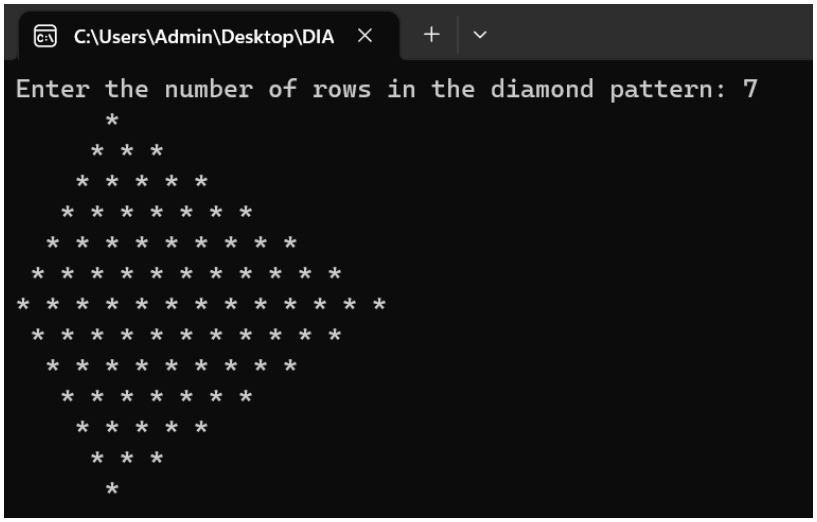


20. Diamond Star Pattern?

INPUT:

```
#include <stdio.h>
void printSpaces(int n) {
   for (int i = 0; i < n; i++) {
      printf(" ");
   }
}
void printStars(int n) {
   for (int i = 0; i < n; i++) {
      printf("* ");
   }
}
int main() {
   int rows, i, j, space;

   printf("Enter the number of rows in the diamond pattern: ");
   scanf("%d", &rows);
   for (i = 1; i <= rows; i++) {
      printSpaces(rows - i);
      printStars(2 * i - 1);
      printf("\n");
   }
   for (i = rows - 1; i >= 1; i--) {
      printSpaces(rows - i);
      printStars(2 * i - 1);
      printf("\n");
   }

   return 0;
}
```

OUTPUT:



21. WRITE C PROGRAM Print the sum of all even numbers between 1 and 100?

INPUT:

```
#include <stdio.h>

int main() {
   int sum = 0;

   for (int i = 2; i <= 100; i += 2) {
      sum += i;
   }

   printf("The sum of all even numbers between 1 and 100 is: %d\n", sum);

   return 0;
}
```

OUTPUT:

```
The sum of all even numbers between 1 and 100 is: 2550

--------------------------------
Process exited after 0.0181 seconds with return value 0
Press any key to continue . . .
```

22. Matrix Multiplication?

INPUT:

```
#include <stdio.h>

#define ROWS1 3
#define COLS1 3
#define ROWS2 3
#define COLS2 3

void multiplyMatrices(int mat1[ROWS1][COLS1], int mat2[ROWS2][COLS2], int
result[ROWS1][COLS2]) {
   for (int i = 0; i < ROWS1; i++) {
      for (int j = 0; j < COLS2; j++) {
         result[i][j] = 0;
         for (int k = 0; k < COLS1; k++) {
            result[i][j] += mat1[i][k] * mat2[k][j];
         }
```

```
        }
    }
}

void displayMatrix(int mat[ROWS1][COLS2]) {
    for (int i = 0; i < ROWS1; i++) {
        for (int j = 0; j < COLS2; j++) {
            printf("%d\t", mat[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int mat1[ROWS1][COLS1] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int mat2[ROWS2][COLS2] = {
        {9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}
    };

    int result[ROWS1][COLS2];

    multiplyMatrices(mat1, mat2, result);

    printf("Matrix 1:\n");
    displayMatrix(mat1);

    printf("\nMatrix 2:\n");
    displayMatrix(mat2);

    printf("\nResultant Matrix:\n");
    displayMatrix(result);

    return 0;
}
```

OUTPUT:

```
Matrix 1:
1       2       3
4       5       6
7       8       9

Matrix 2:
9       8       7
6       5       4
3       2       1

Resultant Matrix:
30      24      18
84      69      54
138     114     90
```

23. Split the Array and Add First Part to the End?

INPUT:

```
#include <stdio.h>
void splitAndAdd(int arr[], int size, int splitIndex) {
   int temp[size];
   for (int i = 0; i < size; i++) {
      temp[i] = arr[i];
   }

   for (int i = 0; i < size; i++) {
      arr[i] = temp[(i + splitIndex) % size] + temp[i];
   }
}

void displayArray(int arr[], int size) {
```

```
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int size, splitIndex;

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

    int arr[size];

    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the split index (0 to %d): ", size - 1);
    scanf("%d", &splitIndex);

    if (splitIndex >= 0 && splitIndex < size) {
        splitAndAdd(arr, size, splitIndex);

        printf("Array after splitting and adding: ");
        displayArray(arr, size);
    } else {
        printf("Invalid split index!\n");
    }
```

```
    return 0;
}
```

OUTPUT:

```
Enter the size of the array: 5
Enter the elements of the array: 1 2 3 4 5
Enter the split index (0 to 4): 2
Array after splitting and adding: 8 10 7 9 6
```

24. Find Union and Intersection of Two Arrays

Enter the elements of Array 1:
INPUT:

```
#include <stdio.h>

void findUnion(int arr1[], int size1, int arr2[], int size2) {
    int unionArr[size1 + size2];
    int i, j, k;


    for (i = 0; i < size1; i++) {
        unionArr[i] = arr1[i];
    }

    for (j = 0; j < size2; j++) {
        int found = 0;
        for (i = 0; i < size1; i++) {
            if (arr2[j] == arr1[i]) {
                found = 1;
                break;
            }
        }
        if (!found) {
            unionArr[size1++] = arr2[j];
        }
    }

    printf("Union of the arrays: ");
    for (i = 0; i < size1; i++) {
```

```
        printf("%d ", unionArr[i]);
    }
    printf("\n");
}

void findIntersection(int arr1[], int size1, int arr2[], int size2) {
    int intersectionArr[size1 < size2 ? size1 : size2];
    int i, j, k = 0;

    for (i = 0; i < size1; i++) {
        for (j = 0; j < size2; j++) {
            if (arr1[i] == arr2[j]) {
                intersectionArr[k++] = arr1[i];
                break;
            }
        }
    }

    printf("Intersection of the arrays: ");
    for (i = 0; i < k; i++) {
        printf("%d ", intersectionArr[i]);
    }
    printf("\n");
}

int main() {
    int size1, size2;

    printf("Enter the number of elements in Array 1: ");
    scanf("%d", &size1);
    int arr1[size1];

    printf("Enter the elements of Array 1: ");
    for (int i = 0; i < size1; i++) {
        scanf("%d", &arr1[i]);
    }

    printf("Enter the number of elements in Array 2: ");
    scanf("%d", &size2);
    int arr2[size2];

    printf("Enter the elements of Array 2: ");
    for (int i = 0; i < size2; i++) {
        scanf("%d", &arr2[i]);
    }
```

```
    findUnion(arr1, size1, arr2, size2);
    findIntersection(arr1, size1, arr2, size2);

    return 0;
}
```

OUTPUT:

```
mathematica

Enter the number of elements in Array 1: 5
Enter the elements of Array 1: 1 2 3 4 5
Enter the number of elements in Array 2: 4
Enter the elements of Array 2: 3 4 5 6
Union of the arrays: 1 2 3 4 5 6
Intersection of the arrays: 3 4 5
```

25. Find Missing Numbers in Array

INPUT:

```
#include <stdio.h>
void findMissingNumbers(int arr[], int size) {
    int i, j;
    int found = 0;
    int last = arr[0];

    printf("Missing numbers in the array: ");
    for (i = 1; i < size; i++) {
        if (arr[i] - last > 1) {
            for (j = last + 1; j < arr[i]; j++) {
                printf("%d ", j);
                found = 1;
            }
        }
        last = arr[i];
    }

    if (!found) {
        printf("None");
    }

    printf("\n");
}

int main() {
```

```
    int size;

    printf("Enter the number of elements in the array: ");
    scanf("%d", &size);

    int arr[size];

    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    findMissingNumbers(arr, size);

    return 0;
}
```

OUTPUT:

```c
Enter the number of elements in the array: 7
Enter the elements of the array: 1 3 5 7 9 12 15
Missing numbers in the array: 2 4 6 8 10 11 13 14
```

26. Print all Non Repeated Elements in an Array

Enter size of the array: 6

INPUT:

```
#include <stdio.h>
void count Occurrences(int arr[], int size, int count[]) {
    for (int i = 0; i < size; i++) {
        count[arr[i]]++;
    }
}
void printNonRepeatedElements(int arr[], int size, int count[]) {
    printf("Non-repeated elements in the array: ");
    for (int i = 0; i < size; i++) {
        if (count[arr[i]] == 1) {
            printf("%d ", arr[i]);
        }
    }
    printf("\n");
```

```
}

int main() {
   int size;

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

   int arr[size];
   int count[100] = {0};

   printf("Enter the elements of the array: ");
   for (int i = 0; i < size; i++) {
      scanf("%d", &arr[i]);
   }

   countOccurrences(arr, size, count);
   printNonRepeatedElements(arr, size, count);

   return 0;
}
```

OUTPUT:

```
c

Enter the size of the array: 6
Enter the elements of the array: 1 2 3 2 4 5
Non-repeated elements in the array: 1 3 4 5
```

27. Find Sum of Array Elements using Pointer

INPUT:

```
#include <stdio.h>
int findSum(int arr[], int size) {
   int sum = 0;
   int *ptr = arr;

   for (int i = 0; i < size; i++) {
      sum += *ptr;
      ptr++;
   }
```

```
    return sum;
}

int main() {
    int size;

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

    int arr[size];

    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    int sum = findSum(arr, size);
    printf("Sum of array elements: %d\n", sum);

    return 0;
}
```

OUTPUT:

```c
Enter the size of the array: 5
Enter the elements of the array: 10 20 30 40 50
Sum of array elements: 150
```

28. Delete an Element from an Array

INPUT:

```
#include <stdio.h>

void deleteElement(int arr[], int size, int position) {

    if (position < 0 || position >= size) {

        printf("Invalid position to delete.\n");
```

```
        return;
    }

    for (int i = position; i < size - 1; i++) {
        arr[i] = arr[i + 1];
    }

    printf("Element at position %d deleted successfully.\n", position);
}

void displayArray(int arr[], int size) {
    printf("Array after deletion: ");
    for (int i = 0; i < size - 1; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int size, position;

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

    int arr[size];

    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
```

```
    printf("Enter the position to delete (0 to %d): ", size - 1);
    scanf("%d", &position);

    deleteElement(arr, size, position);
    displayArray(arr, size);

    return 0;
}
```

OUTPUT:

```mathematica
Enter the size of the array: 6
Enter the elements of the array: 10 20 30 40 50 60
Enter the position to delete (0 to 5): 2
Element at position 2 deleted successfully.
Array after deletion: 10 20 40 50 60
```

29. Find the Number of Elements in an Array

INPUT:

```
#include <stdio.h>

int main() {
   int arr[] = {10, 20, 30, 40, 50};
   int numElements = sizeof(arr) / sizeof(arr[0]);

   printf("The number of elements in the array is: %d\n", numElements);

   return 0;
}
```

OUTPUT:

```c
The number of elements in the array is: 5
```

30. Given an array of integers, find the longest increasing subarray.

INPUT:

```
#include <stdio.h>

void findLongestIncreasingSubarray(int arr[], int size) {
   int longestStart = 0;
   int longestLength = 1;
   int currentStart = 0;
   int currentLength = 1;

   for (int i = 1; i < size; i++) {
      if (arr[i] > arr[i - 1]) {
         currentLength++;
         if (currentLength > longestLength) {
            longestStart = currentStart;
            longestLength = currentLength;
         }
      } else {
         currentStart = i;
         currentLength = 1;
      }
   }
```

```
    printf("Longest increasing subarray: ");
    for (int i = longestStart; i < longestStart + longestLength; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int size;

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

    int arr[size];

    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    findLongestIncreasingSubarray(arr, size);

    return 0;
}
```

OUTPUT:

```
mathematica

Enter the size of the array: 8
Enter the elements of the array: 1 2 3 1 2 3 4 5
Longest increasing subarray: 1 2 3 4 5
```

31. Given an array of integers, find the element that appears more than n/2 times (where n is the size of the array).

INPUT:

```
#include <stdio.h>


int findMajorityElement(int arr[], int size) {

    int majorityElement = arr[0];
```

```
  int count = 1;

   for (int i = 1; i < size; i++) {
      if (arr[i] == majorityElement) {
         count++;
      } else {
         count--;
         if (count == 0) {
            majorityElement = arr[i];
            count = 1;
         }
      }
   }

   // Verify if the majority element appears more than n/2 times
   count = 0;
   for (int i = 0; i < size; i++) {
      if (arr[i] == majorityElement) {
         count++;
      }
   }

   if (count > size / 2) {
      return majorityElement;
   }

   return -1; // No majority element found
}

int main() {
```

```
    int size;

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

    int arr[size];

    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    int majorityElement = findMajorityElement(arr, size);

    if (majorityElement != -1) {
        printf("The majority element is: %d\n", majorityElement);
    } else {
        printf("No majority element found.\n");
    }

    return 0;
}
```

OUTPUT:

```yaml
Enter the size of the array: 7
Enter the elements of the array: 3 2 3 4 3 5 3
The majority element is: 3
```

32. Given an array of integers, rearrange the elements in such a way that all the negative elements come before the positive elements

INPUT:

```
#include <stdio.h>
void swap(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}

void rearrangeNegativePositive(int arr[], int size) {
   int left = 0;      // Index for the leftmost element
   int right = size - 1;  // Index for the rightmost element

   while (left <= right) {
      while (left <= right && arr[left] < 0) {
         left++;
 }
      while (left <= right && arr[right] >= 0) {
         right--;
      }
      if (left <= right) {
         swap(&arr[left], &arr[right]);
         left++;
         right--;
      }
   }
}

void displayArray(int arr[], int size) {
```

```
    printf("Rearranged array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    rearrangeNegativePositive(arr, size);
    displayArray(arr, size);
    return 0;
}
```

OUTPUT:

```c
Enter the size of the array: 10
Enter the elements of the array: -5 10 -7 4 -3 8 -1 0 -2 6
Rearranged array: -5 -7 -3 -1 -2 10 4 8 0 6
```

33. Given an array of integers, find the majority element (an element that appears more than n/2 times, where n is the size of the array) if it exist.

INPUT:

```
#include <stdio.h>


int find_majority_element(int nums[], int n) {
   int count = 0;
   int majority_element = 0;

   for (int i = 0; i < n; i++) {
      if (count == 0) {
         majority_element = nums[i];
      }
      count += (nums[i] == majority_element) ? 1 : -1;
   }



   int majority_count = 0;
   for (int i = 0; i < n; i++) {
      if (nums[i] == majority_element) {
         majority_count++;
      }
   }

   if (majority_count > n / 2) {
      return majority_element;
   } else {
      return -1; // No majority element exists
   }
}
```

```
int main() {
    int arr[] = {3, 3, 4, 2, 4, 4, 2, 4, 4};
    int n = sizeof(arr) / sizeof(arr[0]);

    int result = find_majority_element(arr, n);
    if (result != -1) {
        printf("Majority element: %d\n", result);
    } else {
        printf("No majority element exists.\n");
    }

    return 0;
}
```

OUTPUT:

```mathematica
Majority element: 4
```

34. Given an array of integers, find the subarray with the largest sum.

INPUT:

```
#include <stdio.h>
void find_largest_sum_subarray(int arr[], int n) {
    int max_so_far = arr[0];
    int max_ending_here = arr[0];
    int start = 0;
    int end = 0;
    int temp_start = 0;

    for (int i = 1; i < n; i++) {
        if (arr[i] > max_ending_here + arr[i]) {
```

```
            max_ending_here = arr[i];
            temp_start = i;
        } else {
            max_ending_here = max_ending_here + arr[i];
        }
        if (max_ending_here > max_so_far) {
            max_so_far = max_ending_here;
            start = temp_start;
            end = i;
        }
    }

    printf("Subarray with the largest sum: ");
    for (int i = start; i <= end; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int arr[] = { -2, 1, -3, 4, -1, 2, 1, -5, 4 };
    int n = sizeof(arr) / sizeof(arr[0]);
    find_largest_sum_subarray(arr, n);
    return 0;
}
```

OUTPUT:

Output

```
/tmp/DwTcQ1GAH1.o
Subarray with the largest sum: 4 -1 2 1
```

35. Given an array of integers, find the smallest missing positive integer.

INPUT:

```
#include <stdio.h>
void swap(int* a, int* b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}
int find_smallest_missing_positive(int arr[], int n) {
   for (int i = 0; i < n; i++) {
      while (arr[i] > 0 && arr[i] <= n && arr[i] != arr[arr[i] - 1]) {
         swap(&arr[i], &arr[arr[i] - 1]);
      }
   }

   for (int i = 0; i < n; i++) {
      if (arr[i] != i + 1) {
         return i + 1;
      }
   }
   return n + 1;
}
int main() {
   int arr[] = {3, 4, -1, 1};
   int n = sizeof(arr) / sizeof(arr[0]);
  int result = find_smallest_missing_positive(arr, n);
   printf("The smallest missing positive integer is: %d\n", result);

   return 0;
}
```

OUTPUT:

```vbnet
The smallest missing positive integer is: 2
```

36. Given an array of integers, rearrange the array in such a way that all the even elements come before the odd element?

INPUT:

```
#include <stdio.h>
void swap(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}
void rearrange(int arr[], int n) {
   int i = 0, j = n - 1;
   while (i < j) {
      while (arr[i] % 2 == 0 && i < j)
         i++;
      while (arr[j] % 2 == 1 && i < j)
         j--;
      if (i < j) {
         swap(&arr[i], &arr[j]);
         i++;
         j--;
      }
   }
}
void printArray(int arr[], int n) {
   for (int i = 0; i < n; i++)
```

```
        printf("%d ", arr[i]);

int main() {
    int arr[] = {12, 34, 45, 9, 8, 90, 3};
    int n = sizeof(arr) / sizeof(arr[0]);

    rearrange(arr, n);
    printArray(arr, n);

    return 0;
}
```

OUTPUT:

Copy code

```
90 8 34 12 9 45 3
```

37. Given an array of integers, find the longest subarray with equal number of 0s and 1s.

INPUT:

```
#include<stdio.h>

#include<stdlib.h>

int findMaxLength(int* nums, int numsSize) {

   int n = numsSize;

   int sumLeft[n];

   int startindex = 0, maxlen = 0;

   int endIndex = -1;

   int min, max;

   int i;


   min = nums[0]; max = nums[0];

   sumLeft[0] = (nums[0] == 0)? -1: 1;

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

      sumLeft[i] = sumLeft[i-1] + ((nums[i] == 0)?

                  -1: 1);


      if (sumLeft[i] < min)

         min = sumLeft[i];

      if (sumLeft[i] > max)

         max = sumLeft[i];

   }


   int hash[max-min+1];


   for (i=0; i<max-min+1; i++)

      hash[i] = -1;
```

```
    for (i=0; i<n; i++) {
        if (sumLeft[i] == 0) {
            maxlen = i+1;
            endIndex = i;
    }
        if (hash[sumLeft[i]-min] == -1)
            hash[sumLeft[i]-min] = i;
        else {
            if ((i - hash[sumLeft[i]-min]) > maxlen) {
                maxlen = i - hash[sumLeft[i]-min];
                endIndex = i;
            }
        }
    }

    printf("Starting index = %d\n",
            endIndex - maxlen + 1);
    printf("Ending index = %d\n", endIndex);

    return maxlen;
}
int main() {
    int arr[] = {1, 0, 0, 1, 0, 1, 1};
    int size = sizeof(arr)/sizeof(arr[0]);

    printf("Length = %d\n",
            findMaxLength(arr, size));
    return 0;
}
```

OUTPUT:

```
Starting index = 1
Ending index = 6
Length = 6
```

38. Given an array of integers, find the smallest missing positive integer?

INPUT:

```
#include<stdio.h>
void swap(int* a, int* b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}
int segregate(int arr[], int size) {
   int j = 0, i;
   for(i = 0; i < size; i++) {
     if (arr[i] <= 0) {
        swap(&arr[i], &arr[j]);
        j++;
     }
   }
   return j;
```

```
}
int findMissingPositive(int arr[], int size) {
  int i;
  for(i = 0; i < size; i++) {
    if(abs(arr[i]) - 1 < size && arr[abs(arr[i]) - 1] > 0)
      arr[abs(arr[i]) - 1] = -arr[abs(arr[i]) - 1];
  }
  for(i = 0; i < size; i++)
    if (arr[i] > 0)
      return i+1;
  return size+1;
}
int findMissing(int arr[], int size) {
  int shift = segregate (arr, size);
  return findMissingPositive(arr+shift, size-shift);
}
int main() {
   int arr[] = {0, 10, 2, -10, -20, 1, 3};
   int arr_size = sizeof(arr)/sizeof(arr[0]);
   int missing = findMissing(arr, arr_size);
   printf("The smallest positive missing number is %d \n", missing);
   return 0;
}
```

OUTPUT:

```csharp
The smallest positive missing number is 4
```

39. Given an array of integers, find the two elements that have the maximum product

INPUT:

```
#include<stdio.h>
void maxProduct(int arr[], int n) {
   if(n < 2) {
      printf("No pairs exists\n");
      return;
   }

   if(n == 2) {
      printf("%d %d\n", arr[0], arr[1]);
      return;
   }
   int posa = 0, posb = 0;
   int nega = 0, negb = 0;
   for(int i = 0; i < n; i++) {
      if(arr[i] > posa) {
         posb = posa;
         posa = arr[i];
      } else if(arr[i] > posb)
         posb = arr[i];
      if(arr[i] < 0 && abs(arr[i]) > abs(nega)) {
         negb = nega;
         nega = arr[i];
      } else if(arr[i] < 0 && abs(arr[i]) > abs(negb))
         negb = arr[i];
   }
```

```
    if(nega*negb > posa*posb)
        printf("Max product pair is {%d, %d}\n", nega, negb);
    else
        printf("Max product pair is {%d, %d}\n", posa, posb);
}

int main() {
    int arr[] = {1, 4, 3, 6, 7, 0};
    int n = sizeof(arr)/sizeof(arr[0]);
    maxProduct(arr, n);
    return 0;
}
```

OUTPUT:

```
Max product pair is {7, 6}
```

40. Given an array of integers, find the subarray with the maximum product

INPUT:

```
#include <stdio.h>
int maxProduct(int* nums, int numsSize){
  int maxVal = nums[0];
   int maxEnding = nums[0], minEnding = nums[0];
   for (int i = 1; i < numsSize; i++) {
      if (nums[i] < 0) {
         int temp = maxEnding;
         maxEnding = max(minEnding * nums[i], nums[i]);
         minEnding = temp * nums[i];
      } else {
         maxEnding = max(maxEnding * nums[i], nums[i]);
         minEnding = min(minEnding * nums[i], nums[i]);
      }
      maxVal = max(maxVal, maxEnding);
   }
   return maxVal;
}
int min(int x, int y) { return y ^ ((x ^ y) & -(x < y));
}
int max(int x, int y) { return x ^ ((x ^ y) & -(x < y));
 }
int main() {
   int arr[] = {1, -2, -3, 0, 7, -8, -2};
   int n = sizeof(arr)/sizeof(arr[0]);
   printf("Maximum Subarray product is %d", maxProduct(arr, n));
   return 0;}
```

OUTPUT:

```csharp
Maximum Subarray product is 112
```

41. Given an array of integers, find the longest subarray with the given sum.

OUTPUT:

```
#include <stdio.h>
void findLongestSubarrayBySum(int s, int arr[], int n) {
  int sum = 0;
  int left = 0;
  int right = 0;
  int len = -1;
  int start = 0;

  while (right < n) {
    sum += arr[right];
    while (left < right && sum > s) {
      sum -= arr[left++];
    }
    if (sum == s && (right - left + 1 > len)) {
      len = right - left + 1;
      start = left;
    }
    right++;
  }

  if (len == -1) {
    printf("No subarray with given sum exists");
  } else {
```

```
        printf("Longest Subarray is from index %d to %d\n", start, start+len-1);
    }
}

int main() {
    int arr[] = {1, 2, 3, 7, 5};
    int n = sizeof(arr) / sizeof(arr[0]);
    int s = 12;
    findLongestSubarrayBySum(s, arr, n);
    return 0;
}
```

OUTPUT:

```vbnet
Longest Subarray is from index 1 to 3
```

42. Write a program to count the number of occurrences of a character in a string using pointers.

INPUT:

```
#include<stdio.h>


int countOccurrences(char * str, char c) {
   int count = 0;
   while(*str != '\0') {
      if(*str == c) {
         count++;
      }
      str++;
   }
   return count;
}


int main() {
   char str[] = "Hello, World!";
   char c = 'l';
   printf("Number of occurrences of '%c' is %d\n", c, countOccurrences(str, c));
   return 0;
}
```

OUTPUT:

```
Number of occurrences of 'l' is 3
```

43. Write a program to recursively solve the Tower of Hanoi problem for n disks

INPUT:

```
#include <stdio.h>
void TowerOfHanoi(int n, char from, char to, char aux) {
   if (n == 1) {
      printf("Move disk 1 from rod %c to rod %c\n", from, to);
      return;
   }
   TowerOfHanoi(n-1, from, aux, to);
   printf("Move disk %d from rod %c to rod %c\n", n, from, to);
   TowerOfHanoi(n-1, aux, to, from);
}
int main() {
   int n = 3;
   TowerOfHanoi(n, 'A', 'C', 'B');
   return 0;
}
```

OUTPUT:

Output

```
/tmp/wkATN89eWK.o
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
```

44. Write a program to recursively calculate the sum of all even numbers in a given range

INPUT:

```
#include <stdio.h>.
int sumEven(int lower, int upper) {
   if (lower > upper) {
      return 0;
   }
   if (lower % 2 != 0) {
      return sumEven(lower + 1, upper);
   }
   return lower + sumEven(lower + 2, upper);
}

int main() {
   int lower, upper;
   printf("Enter the lower bound: ");
   scanf("%d", &lower);
   printf("Enter the upper bound: ");
   scanf("%d", &upper);
 int result = sumEven(lower, upper);
   printf("The sum of even numbers from %d to %d is: %d\n", lower, upper, result);
  return 0;
}
```

OUTPUT:

```
Enter the lower bound: 1
Enter the upper bound: 10
The sum of even numbers from 1 to 10 is: 30
```

45. Create a structure named "Employee" to store employee details such as name, employee ID, and salary. Write a program to initialize and display the details of an employee using this structure I

INPUT:

```
#include <stdio.h>

typedef struct {

   char name[100];

   int id;

   float salary;

} Employee;


int main() {

   Employee emp1 = {"John Doe", 1234, 75000.0};

   printf("Employee Details:\n");

   printf("Name: %s\n", emp1.name);

   printf("ID: %d\n", emp1.id);

   printf("Salary: %.2f\n", emp1.salary);


   return 0;

}
```

OUTPUT:

```
Employee Details:
Name: John Doe
ID: 1234
Salary: 75000.00
```

46. Define a structure named "Point" to represent a point in a 2D coordinate system. Write a program to calculate the distance between two points using this structure

INPUT:

```
#include <stdio.h>

#include <math.h>

typedef struct {

   float x;

   float y;

} Point;

float distance(Point p1, Point p2) {

   return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));

}

int main() {

   Point p1 = {1.0, 2.0};

   Point p2 = {4.0, 6.0};

   float dist = distance(p1, p2);

   printf("The distance between the two points is: %.2f\n", dist);

return 0;

}
```

OUTPUT:

```
The distance between the two points is: 5.00
```

47. Create a structure named "Book" to store book details such as title, author, and price. Write a program to initialize an array of books using this structure and display their details.

INPUT:

```
#include <stdio.h>

typedef struct {

   char title[100];

   char author[100];
```

```
    float price;
} Book;

void displayBook(Book book) {
    printf("Title: %s\n", book.title);
    printf("Author: %s\n", book.author);
    printf("Price: %.2f\n", book.price);
    printf("------------------------\n");
}
int main() {
    Book books[3] = {
        {"The Great Gatsby", "F. Scott Fitzgerald", 10.99},
        {"To Kill a Mockingbird", "Harper Lee", 8.99},
        {"1984", "George Orwell", 9.99}
    }
    for (int i = 0; i < 3; i++) {
        displayBook(books[i]);
    }
    return 0;
}
```

OUTPUT:

```
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Price: 10.99
------------------------
Title: To Kill a Mockingbird
Author: Harper Lee
Price: 8.99
------------------------
Title: 1984
Author: George Orwell
Price: 9.99
```

48. Define a structure named "Date" to represent a date (day, month, and year). Write a program to compare two dates using this structure and display which date comes first.

INPUT:

```
#include <stdio.h>
typedef struct {
   int day;
   int month;
   int year;
} Date;
int compareDates(Date d1, Date d2) {
   if (d1.year < d2.year) {
      return -1;
   } else if (d1.year > d2.year) {
      return 1;
   } else {
      if (d1.month < d2.month) {
         return -1;
      } else if (d1.month > d2.month) {
         return 1;
      } else {
         if (d1.day < d2.day) {
            return -1;
         } else if (d1.day > d2.day) {
            return 1;
         } else {
            return 0;
         }
      }
   }
}
```

```
void printDate(Date d) {
   printf("%02d/%02d/%04d\n", d.day, d.month, d.year);
}

int main() {
   Date d1 = {10, 12, 2022};
   Date d2 = {5, 10, 2023};
   int result = compareDates(d1, d2);
   if (result < 0) {
      printDate(d1);
      printf("comes before ");
      printDate(d2);
   } else if (result > 0) {
      printDate(d2);
      printf("comes before ");
      printDate(d1);
   } else {
      printf("The dates are the same.\n");
   }

   return 0;
}
```

OUTPUT:

plaintext

```
10/12/2022
comes before
05/10/2023
```

49. Create a structure named "Student" to store student details such as name, roll number, and marks in three subjects. Write a program to calculate the average marks of a student using this structure.

INPUT:

```
#include <stdio.h>
typedef struct {
   char name[100];
   int rollNo;
   float marks[3];
} Student;
float calculateAverage(Student s) {
   float sum = 0;
   for (int i = 0; i < 3; i++) {
      sum += s.marks[i];
   }
   return sum / 3;
}

int main() {
   Student s1 = {"John Doe", 123, {85.5, 90.0, 88.0};
}
   float average = calculateAverage(s1);
   printf("%s's average marks are: %.2f\n", s1.name, average);

   return 0;
}
```

OUTPUT:

```
John Doe's average marks are: 87.83
```

50. Write a program to recursively calculate the number of ways to reach a target sum using a set of given numbers.

INTPUT:

```
#include <stdio.h>
int numberOfWays(int arr[], int size, int target) {
   if (target == 0)
      return 1;
   if (target < 0)
      return 0;
   if (size <= 0 && target >= 1)
      return 0;
return numberOfWays(arr, size - 1, target) + numberOfWays(arr, size, target - arr[size - 1]);
}int main() {
   int arr[] = {1, 2, 3};
   int size = sizeof(arr) / sizeof(arr[0]);
   int target = 4;
  int result = numberOfWays(arr, size, target);
   printf("The number of ways to reach the target sum is: %d\n", result);
  return 0;
}
```

OUTPUT:

```
he number of ways to reach the target sum is: 4
```

is the output you should see in the terminal or console wh

the provided program. It calculates and prints the number