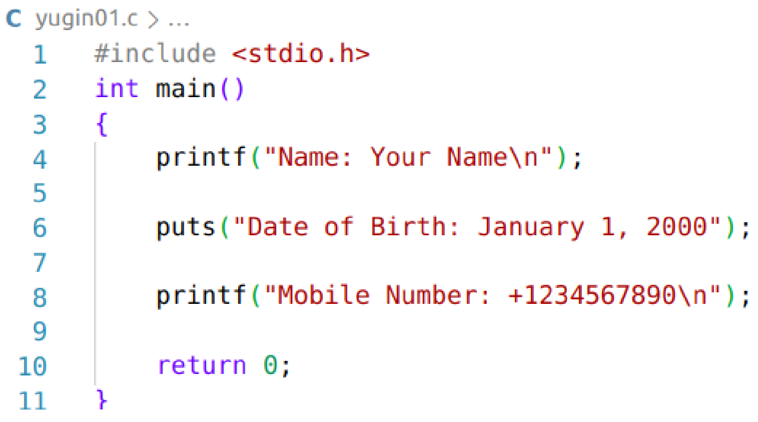
1. Write a C program to print your name, date of birth and mobile number using printf() and puts() functions.

#Algorithm:
Step 1: Start
Step 2: Print "Name: Your Name" using printf().
Step 3: Print "Date of Birth: January 1, 2000" using puts().
Step 4: Print "Mobile Number: +1234567890" using printf().
Step 5: End.

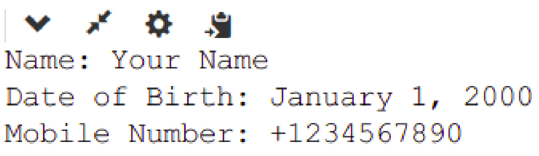
#Flowchart:

#Program:



```
yugin01.c > ...
#include <stdio.h>
int main()
{
    printf("Name: Your Name\n");

    puts("Date of Birth: January 1, 2000");

    printf("Mobile Number: +1234567890\n");

    return 0;
}
```

#Output:



```
Name: Your Name
Date of Birth: January 1, 2000
Mobile Number: +1234567890
```

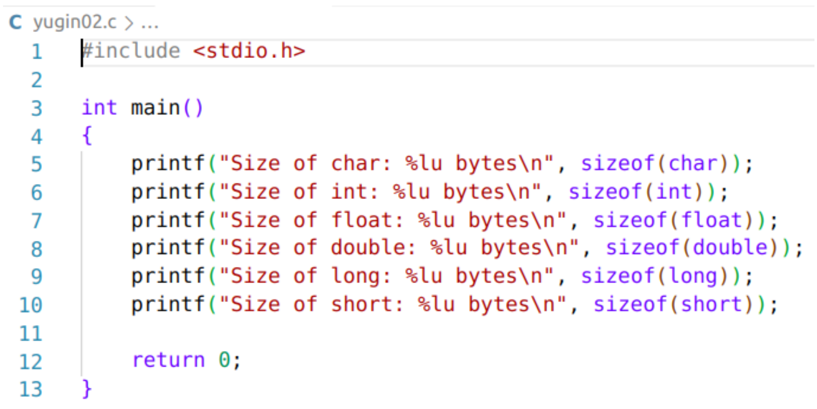
2. Write a C program to display size in bytes of different data types using sizeof() operator.

#Algorithm:

```
Step 1: Start
Step 2: Print "Size of char: " followed by the size in bytes of char using sizeof( ) and %lu format
        specifier.
Step 3: Print "Size of int: " followed by the size in bytes of int using sizeof( ) and %lu format
        specifier.
Step 4: Print "Size of float: " followed by the size in bytes of float using sizeof( ) and %lu format
        specifier.
Step 5:Print "Size of double: " followed by the size in bytes of double using sizeof( ) and %lu
        format specifier.
Step 6: Print "Size of long: " followed by the size in bytes of long using sizeof( ) and %lu format
        specifier.
Step 7: Print "Size of short: " followed by the size in bytes of short using sizeof( ) and %lu format
        specifier.
Step 8: End.
```

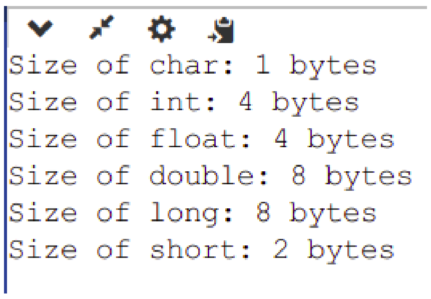
#Flowchart

#Program:


C yugin02.c > ...

```c
#include <stdio.h>

int main()
{
    printf("Size of char: %lu bytes\n", sizeof(char));
    printf("Size of int: %lu bytes\n", sizeof(int));
    printf("Size of float: %lu bytes\n", sizeof(float));
    printf("Size of double: %lu bytes\n", sizeof(double));
    printf("Size of long: %lu bytes\n", sizeof(long));
    printf("Size of short: %lu bytes\n", sizeof(short));

    return 0;
}
```

#Output:



```
Size of char: 1 bytes
Size of int: 4 bytes
Size of float: 4 bytes
Size of double: 8 bytes
Size of long: 8 bytes
Size of short: 2 bytes
```

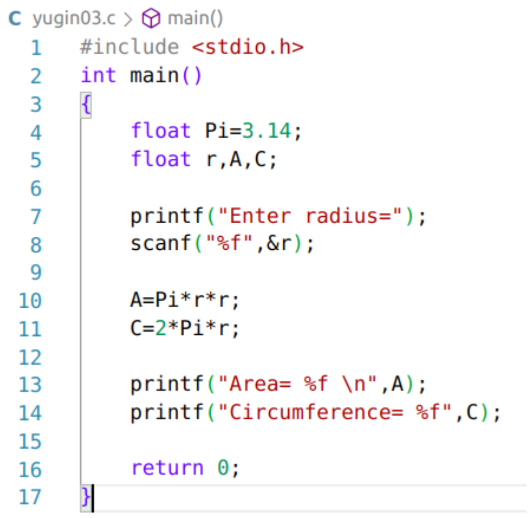
3. Write algorithm, flow-chart and program to compute the area and circumference of a circle with given radius r as input defining π as constant (Note: Area¿πr^2)

#Algorithm:

Step 1: Start.
Srep 2: Declare constant Pi with the value 3.14159.
Step 3: Input radius r.
Step 4: Calculate area using the formula : area = Pi*r *r.
Step 5: Calculate circumference using the formula : circumference = 2*Pi*r.
Step 6: Output the calculated area and circumference
Step 7: End

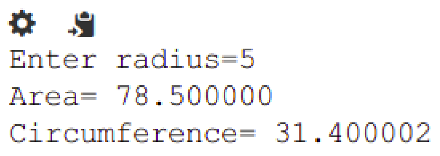
#Flowchart:

#Program:



```c
#include <stdio.h>
int main()
{
    float Pi=3.14;
    float r,A,C;

    printf("Enter radius=");
    scanf("%f",&r);

    A=Pi*r*r;
    C=2*Pi*r;

    printf("Area= %f \n",A);
    printf("Circumference= %f",C);

    return 0;
}
```

#Output:



```
Enter radius=5
Area= 78.500000
Circumference= 31.400002
```

4. Write a C program to convert specified no of days into years, weeks and days. ***(Note: Ignore leap year.)***

#Algorithm:

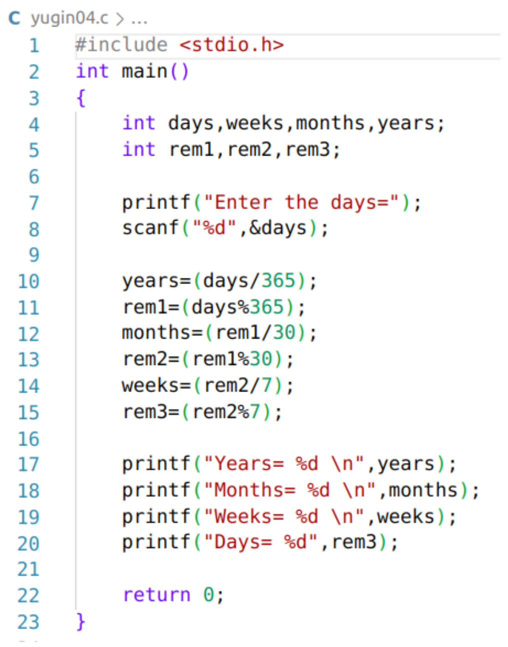
Step 1: Start
Step 2: Input the total number of days.(days)
Step 3: Calculate the number of years: (years= days/365)
Step 4: Calculate the remaining days after extracting the years: (rem1= days%365)
Step 5: Calculate the number of months based on the remaining days (approximating each month to have 30 days): (months= rem1/30)
Step 6: Calculate the remaining days after extracting the months: (rem2=rem1%30)
Step 7: Calculate the number of weeks: (weeks=rem2/7)
Step 8: Calculate the remaining days after extracting the weeks: (rem3=rem2%7)
Step 9: Output the result as:

- Years= years
- Months= months
- Weeks= weeks
- Days= rem3

Step 10: End

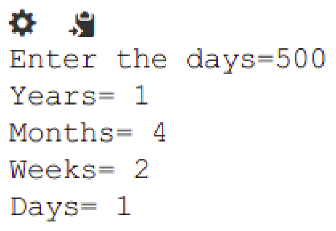
#Flowchart:

#Program:



```c
#include <stdio.h>
int main()
{
    int days,weeks,months,years;
    int rem1,rem2,rem3;

    printf("Enter the days=");
    scanf("%d",&days);

    years=(days/365);
    rem1=(days%365);
    months=(rem1/30);
    rem2=(rem1%30);
    weeks=(rem2/7);
    rem3=(rem2%7);

    printf("Years= %d \n",years);
    printf("Months= %d \n",months);
    printf("Weeks= %d \n",weeks);
    printf("Days= %d",rem3);

    return 0;
}
```

#Output:



```
Enter the days=500
Years= 1
Months= 4
Weeks= 2
Days= 1
```

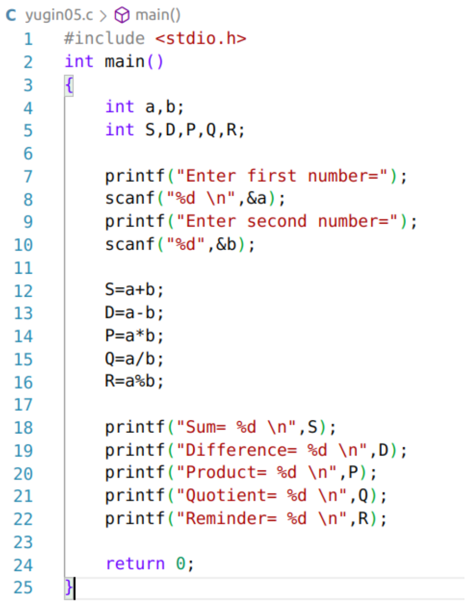
5. Write an algorithm and C program that accepts two integers from the user as input and calculates the sum, difference, product, quotient and remainder applying different arithmetic operators between two integers.

<u>#Algorithm:</u>

```
Step 1: Start
Step 2: Declare variable a,b,S,D,P,Q and R as integers.
Step 2: Input the integer a and b.
Step 3: Calculate the sum: ( S=a+b )
Step 4: Calculate the difference: ( D=a-b )
Step 5: Calculate the product: ( P=a*b )
Step 6: Calculate the quotient: ( Q=a/b )
Step 7: Calculate the remainder: ( R=a%b )
Step 8: Output the results:
    • Print "Sum: S"
    • Print "Difference:D"
    • Print "Product:P"
    • Print "Quotient:Q"
    • Print "Remainder:R"
Step 9: End
```

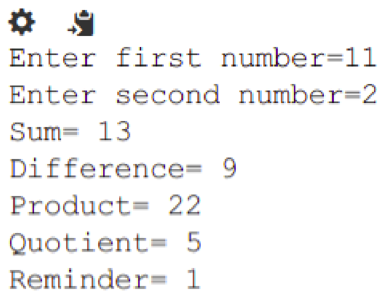
<u>#Flowchart:</u>

<u>#Program:</u>



```c
#include <stdio.h>
int main()
{
    int a,b;
    int S,D,P,Q,R;

    printf("Enter first number=");
    scanf("%d \n",&a);
    printf("Enter second number=");
    scanf("%d",&b);

    S=a+b;
    D=a-b;
    P=a*b;
    Q=a/b;
    R=a%b;

    printf("Sum= %d \n",S);
    printf("Difference= %d \n",D);
    printf("Product= %d \n",P);
    printf("Quotient= %d \n",Q);
    printf("Reminder= %d \n",R);

    return 0;
}
```

<u>#Output:</u>



```
Enter first number=11
Enter second number=2
Sum= 13
Difference= 9
Product= 22
Quotient= 5
Reminder= 1
```

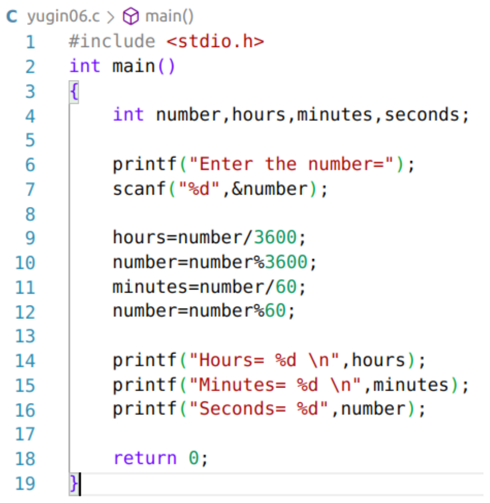
6. Write a C program to convert a given integer (in seconds) to hours, minutes and seconds.

#Algorithm:

```
Step 1: Start
Step 2: Input the integer number. ( number )
Step 3: Divide the input number by 3600 and store the quotient in the variable hours:
        ( hours=number/3600 )
Step 4: Update the input number to store the remaining seconds after extracting hours:
        ( number=number%3600 )
Step 5: Divide the updated number by 60 store the quotient in the variable minutes:
        ( minutes=number/60 )
Step 6: Update the input number to store the remaining seconds after extracting hours:
        ( number=number%60 )
Step 7: Output the results:
            • Hours=hours
            • Minutes=minutes
            • Seconds=number
Step 8: End
```

#Flowchart

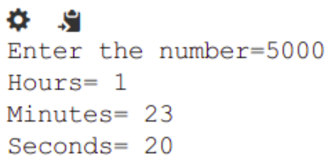
#Program



yugin06.c > main()

```c
#include <stdio.h>
int main()
{
    int number,hours,minutes,seconds;

    printf("Enter the number=");
    scanf("%d",&number);

    hours=number/3600;
    number=number%3600;
    minutes=number/60;
    number=number%60;

    printf("Hours= %d \n",hours);
    printf("Minutes= %d \n",minutes);
    printf("Seconds= %d",number);

    return 0;
}
```

#Output:



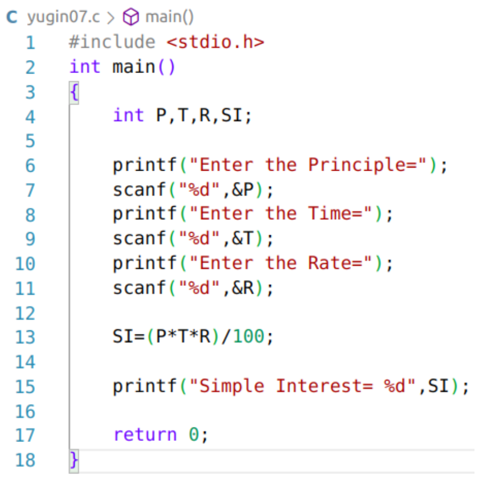
```
Enter the number=5000
Hours= 1
Minutes= 23
Seconds= 20
```

7. Write a C program that accepts principle, rate of interest, time in years and computes the simple interest.

#Algorithm:
Step 1: Start
Step 2: Inputs for the principle amount (P), rate of interest per annum (R), and time in years (T).
Step 3: Calculate the simple interest: [SI=(P*T*R)/100]
Step 4: Printf the simple interest.
Step 5: End

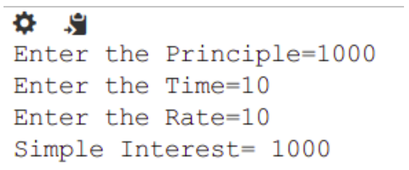
#Flowchart

#Program



```c
#include <stdio.h>
int main()
{
    int P,T,R,SI;

    printf("Enter the Principle=");
    scanf("%d",&P);
    printf("Enter the Time=");
    scanf("%d",&T);
    printf("Enter the Rate=");
    scanf("%d",&R);

    SI=(P*T*R)/100;

    printf("Simple Interest= %d",SI);

    return 0;
}
```

#Output:



```
Enter the Principle=1000
Enter the Time=10
Enter the Rate=10
Simple Interest= 1000
```

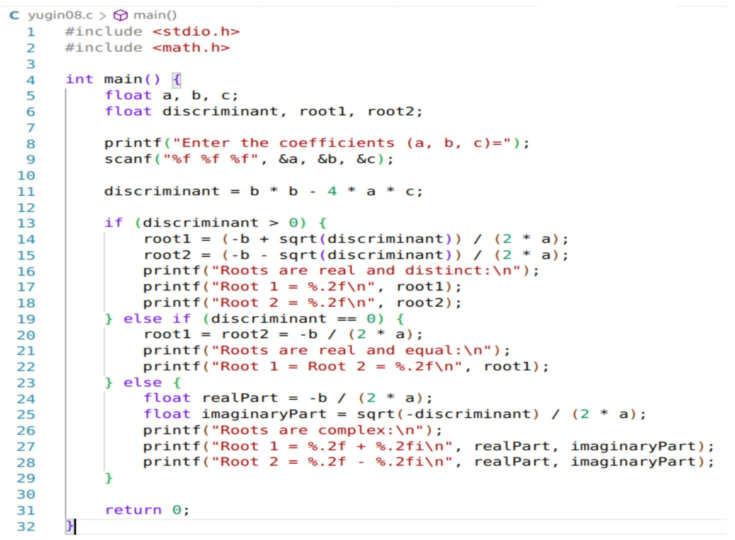
8. Write algorithm pseudo-code as well as draw flow chart to Compute the roots of the quadratic equation $ax^2+bx+c=0$ for given coefficient input a, b and c. Write C program.

#Algorithm:

```
Step 1: Start
Step 2: Take input for coefficients a,b and c.
Step 3: Calculate the discriminant: ( D=b*b-4*c*a )
Step 4: If D>0:
        • Calculate two real and distinct roots: [ root1=(-b+ $\sqrt{D}$ )/(2*a) and
          root2=(-b- $\sqrt{D}$ )/(2*a) ].
        • Print the roots.
Step 5: If D==0:
    • Calculate a real and equal roots: [ root1=root2=(-b)/(2*a) ].
    • print the root.
Step 6: Else:
    • Calculate two complex roots using the real and imaginary parts: [ real root=(-b)/(2*a) and
      imaginary root= $\sqrt{|D|}$ /(2*a) ].
    • Print the complex root.
Step 7: End
```

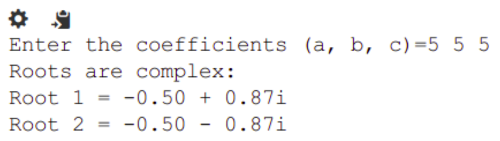
#Flowchart:

#Program:



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

int main() {
    float a, b, c;
    float discriminant, root1, root2;

    printf("Enter the coefficients (a, b, c)=");
    scanf("%f %f %f", &a, &b, &c);

    discriminant = b * b - 4 * a * c;

    if (discriminant > 0) {
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("Roots are real and distinct:\n");
        printf("Root 1 = %.2f\n", root1);
        printf("Root 2 = %.2f\n", root2);
    } else if (discriminant == 0) {
        root1 = root2 = -b / (2 * a);
        printf("Roots are real and equal:\n");
        printf("Root 1 = Root 2 = %.2f\n", root1);
    } else {
        float realPart = -b / (2 * a);
        float imaginaryPart = sqrt(-discriminant) / (2 * a);
        printf("Roots are complex:\n");
        printf("Root 1 = %.2f + %.2fi\n", realPart, imaginaryPart);
        printf("Root 2 = %.2f - %.2fi\n", realPart, imaginaryPart);
    }

    return 0;
}
```

#Output:



```
Enter the coefficients (a, b, c)=5 5 5
Roots are complex:
Root 1 = -0.50 + 0.87i
Root 2 = -0.50 - 0.87i
```

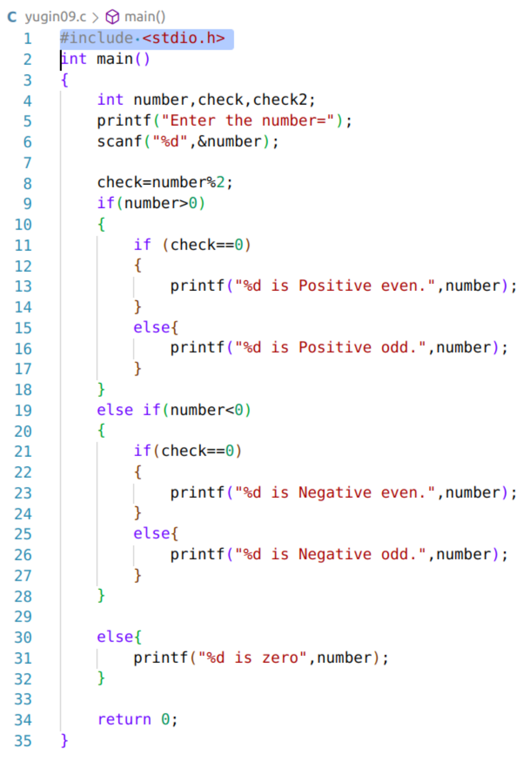
9. Write a C program to check a given integer is positive even, negative even, positive odd or negative odd.

#Algorithm:

```
Step 1: Start
Step 2: Input the number.
Step 3: Calculate: ( check=number%2 )
Step 4: If (number>0)
        • Check if check is equal to 0. (check==0)
        • If true print "The number is positive even" else print "The number is positive odd".
Step 5: Else if(number<0)
        • Check if check is equal to 0.(check==0)
        • If true print "The number is negative even" else print "The number is negative odd".
Step 6: Else print "Te nnumber is 0".
Step 7: End
```

#Flowchart:

#Program:



yugin09.c > main()

```c
#include <stdio.h>
int main()
{
    int number,check,check2;
    printf("Enter the number=");
    scanf("%d",&number);

    check=number%2;
    if(number>0)
    {
        if (check==0)
        {
            printf("%d is Positive even.",number);
        }
        else{
            printf("%d is Positive odd.",number);
        }
    }
    else if(number<0)
    {
        if(check==0)
        {
            printf("%d is Negative even.",number);
        }
        else{
            printf("%d is Negative odd.",number);
        }
    }

    else{
        printf("%d is zero",number);
    }

    return 0;
}
```

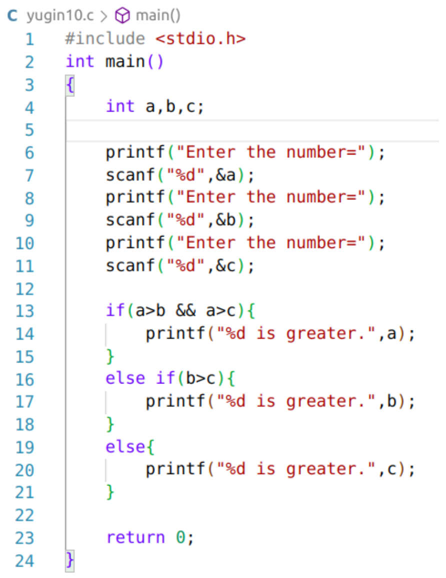
10. Draw a flow chart and write a C program that accepts three integers as input and find the largest of three.

#Algorithm:

```
Step 1: Start
Step 2: Input 3 numbers a,b and c.
Step 3: If (a>b) and (a>c) print “A in greater.”
Step 4: Else if (b>c) print “B is greater.”
Step 5: Else print “C is greater.”
Step 6: End
```

#Flowchart:

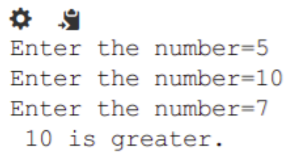
#Program:



yugin10.c > main()

```c
#include <stdio.h>
int main()
{
    int a,b,c;

    printf("Enter the number=");
    scanf("%d",&a);
    printf("Enter the number=");
    scanf("%d",&b);
    printf("Enter the number=");
    scanf("%d",&c);

    if(a>b && a>c){
        printf("%d is greater.",a);
    }
    else if(b>c){
        printf("%d is greater.",b);
    }
    else{
        printf("%d is greater.",c);
    }

    return 0;
}
```

#Output:



```
Enter the number=5
Enter the number=10
Enter the number=7
 10 is greater.
```

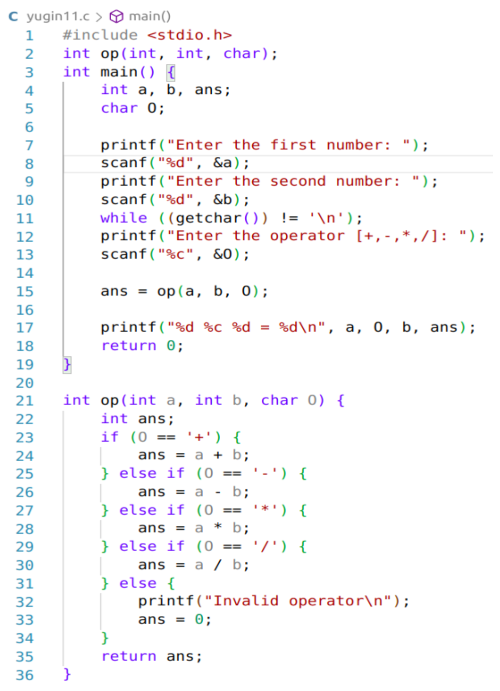
11. Write a program that takes input of two numbers and an operator in(+,-,*,/) as input and pass those numbers and an operator to the function. The function should calculate the result of two numbers as indicated by operator and return the result. Display the result of computation in your program.

#Algorithm:

```
Step 1: Start
Step 2: Input 2 integers a and b and a operator sign (+,-,*,/);
Step 3:Call the function op with arguments a,b and O. Store the result in the variable ans.
        [ Go to Step:5 ]
Step 4: Display the result of the operation (ans) along with the entered numbers and operator.
        [ Go To Step: 9]
Step 5: Define a function op that takes three parameters (a,b and O).
Step 6: Declare a local variable (ans) of type integer.
Step 7: Check Operator and Perform Operation:
        •  if O is '+' ans = a + b
        •  else if O is '-' ans = a - b
        •  else if O is '*' ans = a * b
        •  else if O is '/' ans = a / b
        •  else print "Invalid operator" ans = 0
Step 8: Return the result (ans) [ Go To Step 4 ]
            Step 9: End
```

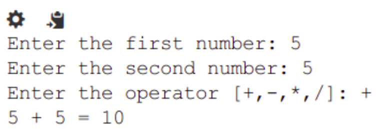
#Flowchart:

#Program:



```c
#include <stdio.h>
int op(int, int, char);
int main() {
    int a, b, ans;
    char O;

    printf("Enter the first number: ");
    scanf("%d", &a);
    printf("Enter the second number: ");
    scanf("%d", &b);
    while ((getchar()) != '\n');
    printf("Enter the operator [+,-,*,/]: ");
    scanf("%c", &O);

    ans = op(a, b, O);

    printf("%d %c %d = %d\n", a, O, b, ans);
    return 0;
}

int op(int a, int b, char O) {
    int ans;
    if (O == '+') {
        ans = a + b;
    } else if (O == '-') {
        ans = a - b;
    } else if (O == '*') {
        ans = a * b;
    } else if (O == '/') {
        ans = a / b;
    } else {
        printf("Invalid operator\n");
        ans = 0;
    }
    return ans;
}
```

#Output:



```
Enter the first number: 5
Enter the second number: 5
Enter the operator [+,-,*,/]: +
5 + 5 = 10
```

12. Write a program to determine whether a given number is palindrome or not.

#Algorithm:

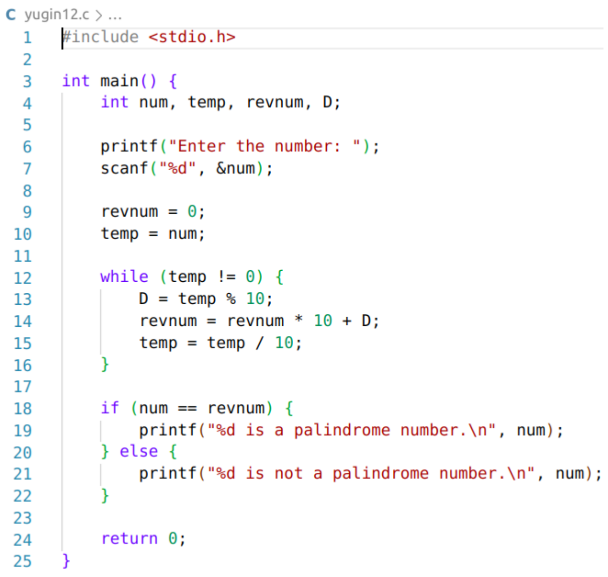
Step 1: Start
Step 2: Input a number(num).
Step 3: Set a variable reverse number to 0(revnum=0) and variable to store original num(temp=num).
Step 4: Start the while loop checking whether (temp!=0). And to the following Operations:

- D=temp%10
- revnum=revnum*10+D
- temp=temp/10

Step 5: If (num==temp) then print “The number is palindrome.” else print “The number is not palindrome.”
Step 6: End

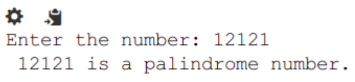
#Flowchart:

#Program:



```
C yugin12.c > ...
#include <stdio.h>

int main() {
    int num, temp, revnum, D;

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

    revnum = 0;
    temp = num;

    while (temp != 0) {
        D = temp % 10;
        revnum = revnum * 10 + D;
        temp = temp / 10;
    }

    if (num == revnum) {
        printf("%d is a palindrome number.\n", num);
    } else {
        printf("%d is not a palindrome number.\n", num);
    }

    return 0;
}
```

#Output:



```
Enter the number: 12121
 12121 is a palindrome number.
```

13. Write a program to determine whether a given number is Armstrong number or not.

#Algorithm:

Step 1: Start
Step 2: Declare variables num,temp,n,result and remof type integer.
Step 3: Input a number(num).
Step 4: Set temp to the value of num(temp=num) and set result and n to 0(resuit=n=0).
Step 5: Start the while loop checking whether (temp!=0). And to the following Operations:

- temp=temp/10
- n+1 or n++

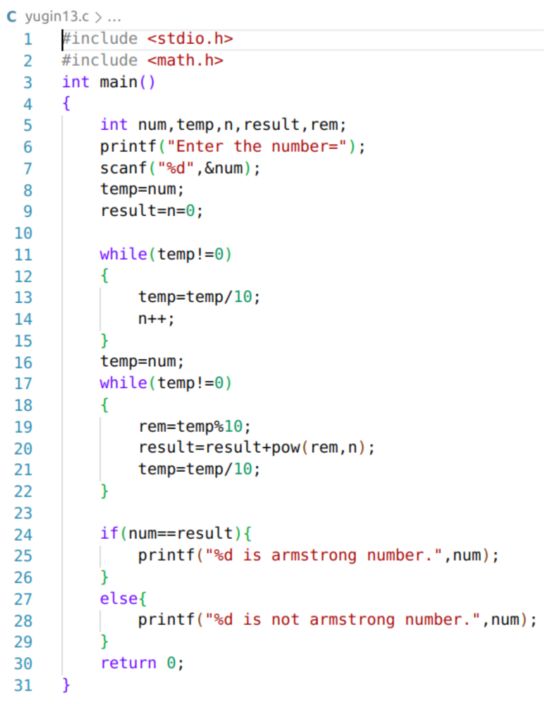
Step 6: Again set temp to the value of num(temp=num).
Step 7: Start the while loop checking whether (temp!=0). And to the following Operations:

- rem=temp%10
- result=result+rem^n
- temp=temp/10

Step 8: If (num==result) print “The number is armstrong.” else print “The number is not arnstrong.”

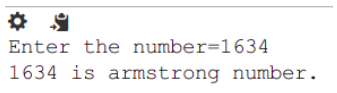
#Flowchart:

#Program:



```c
#include <stdio.h>
#include <math.h>
int main()
{
    int num,temp,n,result,rem;
    printf("Enter the number=");
    scanf("%d",&num);
    temp=num;
    result=n=0;

    while(temp!=0)
    {
        temp=temp/10;
        n++;
    }
    temp=num;
    while(temp!=0)
    {
        rem=temp%10;
        result=result+pow(rem,n);
        temp=temp/10;
    }

    if(num==result){
        printf("%d is armstrong number.",num);
    }
    else{
        printf("%d is not armstrong number.",num);
    }
    return 0;
}
```

#Output:



```
Enter the number=1634
1634 is armstrong number.
```

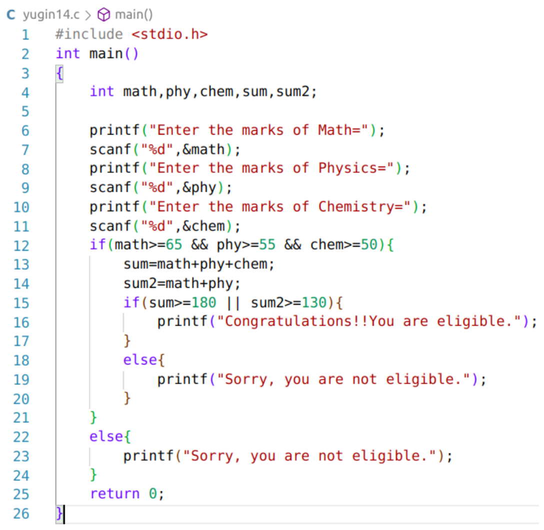
14. Write a C program to find the eligibility of admission for a professional course based on the following criteria:[Marks in Maths >=65, Marks in Phy >=55, Marks in Chem>=50] And Total in all three subject >=180 or Total in Math and physics Subjects >=130

#Algorithm:

```
Step 1: Start
Step 2: Declare the variables math,phy,chem,sum and sum2;
Step 3: If (math>=65 && phy>=55 && chem>=50) [ Go To Step 4 ] else [ Go To Step 6 ]
Step 4: Calculate sum and sum2:
        • sum= math+phy+chem
        • sum2= math+phy
Step 5: If (sum>=180 || sum2>=130) print "You are eligible." else print "Your are not eligible."
        [ Go To Step 7 ]
Step 6: Print "Your are not eligible."
Sep 7: End
```

#Flowchart:

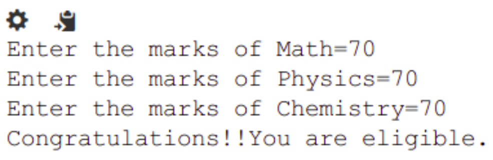
#Program:



yugin14.c > main()

```c
#include <stdio.h>
int main()
{
    int math,phy,chem,sum,sum2;

    printf("Enter the marks of Math=");
    scanf("%d",&math);
    printf("Enter the marks of Physics=");
    scanf("%d",&phy);
    printf("Enter the marks of Chemistry=");
    scanf("%d",&chem);
    if(math>=65 && phy>=55 && chem>=50){
        sum=math+phy+chem;
        sum2=math+phy;
        if(sum>=180 || sum2>=130){
            printf("Congratulations!!You are eligible.");
        }
        else{
            printf("Sorry, you are not eligible.");
        }
    }
    else{
        printf("Sorry, you are not eligible.");
    }
    return 0;
}
```

#Output:



```
Enter the marks of Math=70
Enter the marks of Physics=70
Enter the marks of Chemistry=70
Congratulations!!You are eligible.
```

15. Write a C program to find the sum of first 100 natural numbers using loop.

#Algorithm:

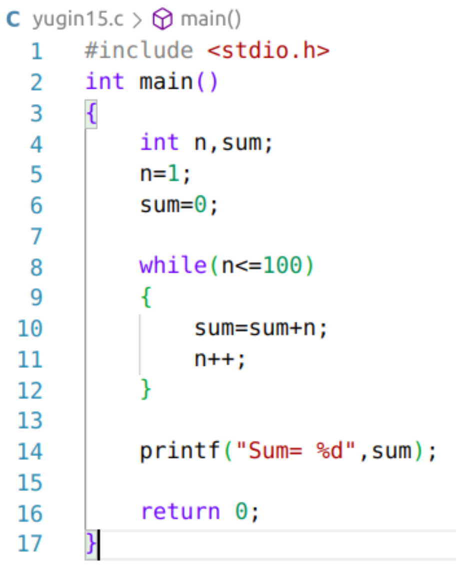
Step 1: Start
Step 2: Declare the variables n and sum.
Step 3: Start the while loop checking whether n is less than equal to 100. And do following operations:

- sum=sum+n
- n+1 or n++

Step 4: Print the Sum.
Step 5: End

#Flowchart:

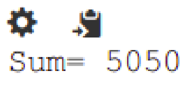
#Program:



yugin15.c > main()

```c
#include <stdio.h>
int main()
{
    int n,sum;
    n=1;
    sum=0;

    while(n<=100)
    {
        sum=sum+n;
        n++;
    }

    printf("Sum= %d",sum);

    return 0;
}
```

#Output:



```
Sum= 5050
```

16. Write a program in C to display the multiplication table of a given integer.

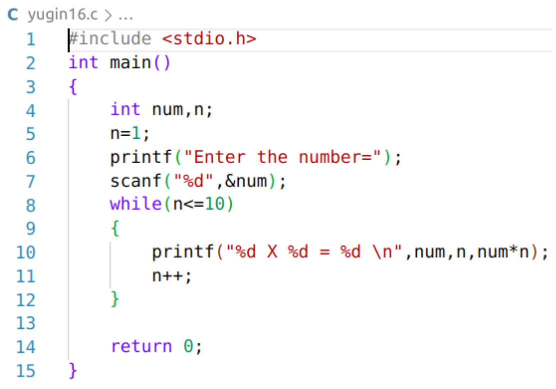
#Algorithm:

Step 1: Start
Step 2: Declare the variables num and n.
Step 3: Input the number (num). Set the n to 1.
Step 4: Start the while loop checking whether n is less than or equal to 10. And do the following operations:
- print “num X n = (num*n)”
- n+1 or n++

Step 5: End

#Flowchart:

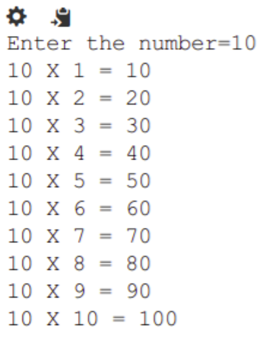
#Program:



C yugin16.c > ...

```c
#include <stdio.h>
int main()
{
    int num,n;
    n=1;
    printf("Enter the number=");
    scanf("%d",&num);
    while(n<=10)
    {
        printf("%d X %d = %d \n",num,n,num*n);
        n++;
    }

    return 0;
}
```

#Output:



```
Enter the number=10
10 X 1 = 10
10 X 2 = 20
10 X 3 = 30
10 X 4 = 40
10 X 5 = 50
10 X 6 = 60
10 X 7 = 70
10 X 8 = 80
10 X 9 = 90
10 X 10 = 100
```

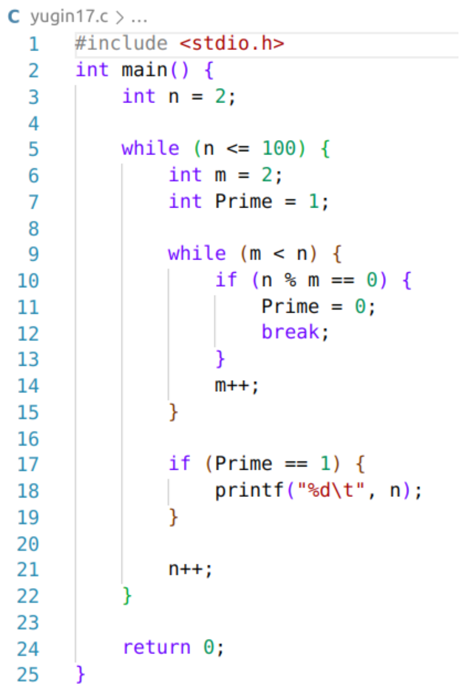
17. Write an algorithm/ program to print the prime numbers up to 100.

#Algorithm:

```
Step 1: Start
Step 2: Declare a variable n and set its value to 2.
Step 3: Start while loop checking whether n is less than or equal to 100. And to following
        operations:
        • Declare m variable and set its value to 2.
        • Declare prime variable and set its value to 1.
        • [ Go To Step 4 ]
        • If (prime==1) then print "n".
        • n+1 or n++
        • [ Go To Step 5 ]
Step 4: Start another while loop inside the first loop checking whether m is less tha n. And do
        following operations:
        • if (n%m == 0) then prime=0 and break the loop
        • m+1 or m++
        • [ Back to Step 3 ]
Step 5: End
```

#Flowchart:

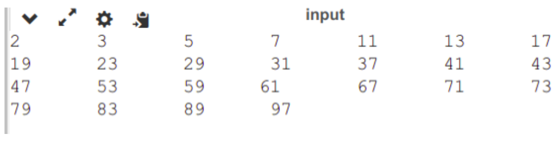
#Program:



C yugin17.c > ...

```c
#include <stdio.h>
int main() {
    int n = 2;

    while (n <= 100) {
        int m = 2;
        int Prime = 1;

        while (m < n) {
            if (n % m == 0) {
                Prime = 0;
                break;
            }
            m++;
        }

        if (Prime == 1) {
            printf("%d\t", n);
        }

        n++;
    }

    return 0;
}
```

#Output:



```
2       3       5       7       11      13      17
19      23      29      31      37      41      43
47      53      59      61      67      71      73
79      83      89      97
```

18. Write algorithm and program to compute the followings using for, do while and while loop separately.
a. factorial of an integer N

#Algorithm:

#Flowchart

#Program

b.computation of a^b(a raised to power b)

#Algorithm:

#Flowchart:

#Program:

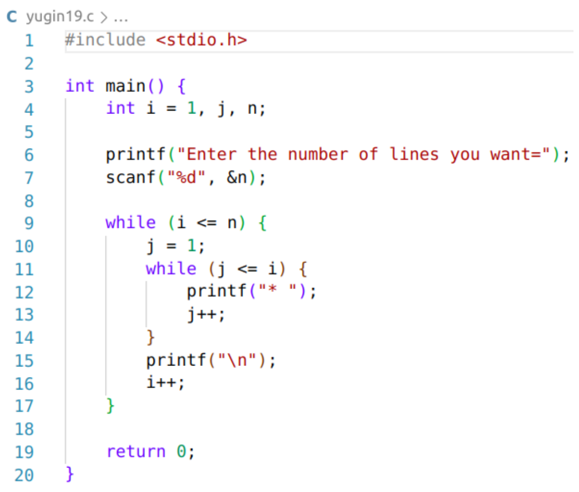
19. Write a program in C to make such a pattern of astrisk(*) below using loop.
*****
*** ***
*** * ***
*** * * *…...n**

#Algorithm:

```
Step 1: Start
Step 2: Declare the variables i,j and n. And set the value of i to 1.
Step 3: Input the value of n.
Step 4: Start the while loop checking whether i is less than or equal to n. And do the following
        operations:
            • Set the value of j to 1.
            • [ Go To Step 5]
            • print "\n" (break or new line)
            • i++ or i+1
            • [Go To Step 6]
Step 5: Start another while loop inside checking whether j is less than or equal to i. And do the
        following operations:
            • print "*"
            • j+1 or j++
            • [ Back to Step 4 ]
Step 6: End
```

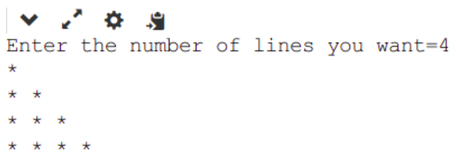
#Flowchart:

#Program:



```c
#include <stdio.h>

int main() {
    int i = 1, j, n;

    printf("Enter the number of lines you want=");
    scanf("%d", &n);

    while (i <= n) {
        j = 1;
        while (j <= i) {
            printf("* ");
            j++;
        }
        printf("\n");
        i++;
    }

    return 0;
}
```

#Output:



```
Enter the number of lines you want=4
*
* *
* * *
* * * *
```

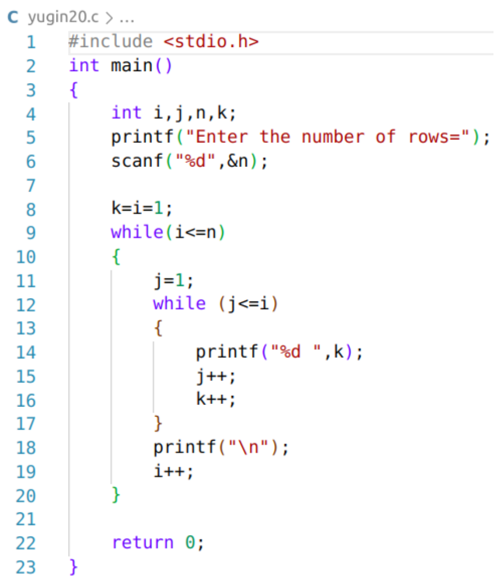
20. Write a program using loop to print the following floyd's triangle as given below when input is n.
1
2 3
4 5 6
7 8 9 10 …. n

#Algorithm:

```
Step 1: Start
Step 2: Declare the variables i,j,n and k. And set the value of k and i to 1.
Step 3: Input the number of rows(n).
Step 4: Start the while loop checking whether i is less tha or equal to n. And do the following
        operations:
            • Set the value of j to 1.
            • [ Go To Step 5 ]
            • Print “\n” (Break or new line)
            • i++ or i+1
            • [ Go To Step 6]
Step 5: Start the while loop checking whether j is less than or equal to i. And do the following
        operations:
            • print the value of k
            • j++ or j+1
            • k++ or k+1
            • [ Back To Step 4]
Step 6: End
```

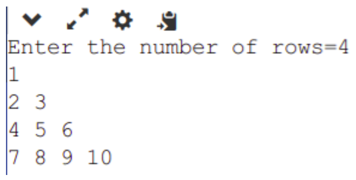
#Flowchart:

#Program:



```c
#include <stdio.h>
int main()
{
    int i,j,n,k;
    printf("Enter the number of rows=");
    scanf("%d",&n);

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

    return 0;
}
```

#Output:



```
Enter the number of rows=4
1
2 3
4 5 6
7 8 9 10
```

21. Write a program in C to make such a pattern like a pyramid with numbers increased by 1.

```
            1
        2   3   4
    5   6   7   8   9
10 11 12 13 14 15 16….n
```

#Algorithm:

Step 1: Start
Step 2: Declare the variables i,j,k,l,m,n and a. Set the value of i,k and a to 1.
Step 3: Input the number of rows(n).
Step 4: Set the value of m to n.(m=n)
Step 5: Start the while loop checking whether i is less than or equal to n. And do the following operations:

- Set the value of l and j to 1.
- [Go To Step 6]
- [Go To Step 7]
- Print "\n" (Break or new line)
- i++, m-- and a+=2 or (a=a+2)
- [Go To Step 8]

Step 6: Start the while loop checking whether l is less than m. And do the following operations:

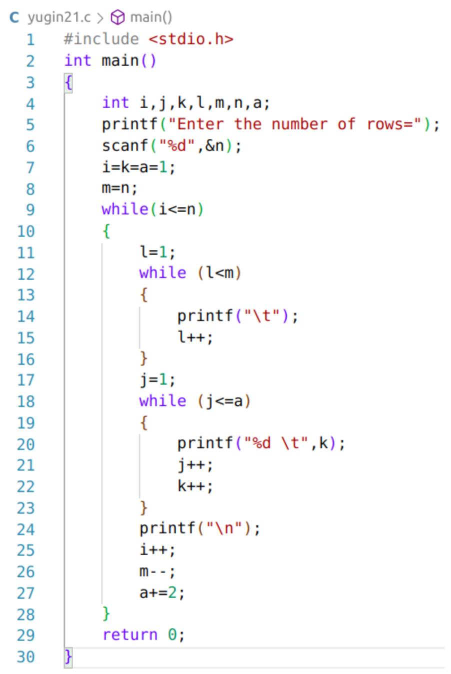
- Print "\t" (tab or space)
- l++ or l+1
- [Back To Step 5]

Step 7: Start the while loop checking whether j is less than or equal to a. And do the following operations:

- Print the value of k.
- j++ and k++
- [Back To Step 5]

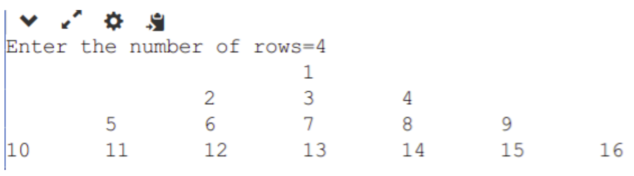
Step 8: End

#Program:



```c
#include <stdio.h>
int main()
{
    int i,j,k,l,m,n,a;
    printf("Enter the number of rows=");
    scanf("%d",&n);
    i=k=a=1;
    m=n;
    while(i<=n)
    {
        l=1;
        while (l<m)
        {
            printf("\t");
            l++;
        }
        j=1;
        while (j<=a)
        {
            printf("%d \t",k);
            j++;
            k++;
        }
        printf("\n");
        i++;
        m--;
        a+=2;
    }
    return 0;
}
```

#Output



```
Enter the number of rows=4
                              1
                    2         3         4
          5         6         7         8         9
10        11        12        13        14        15        16
```

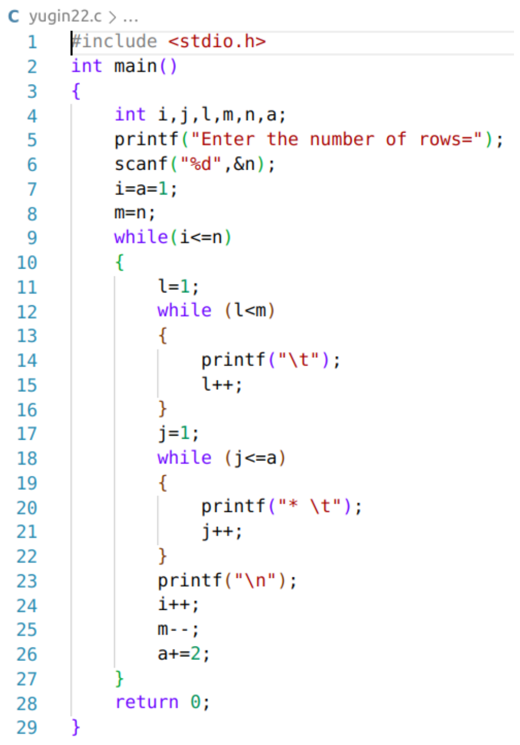
22. Write a program in C to make such a pattern like a pyramid with an asterisk.

```
      *
    * * *
  * * * * *
* * * * * * * …..n
```

#Algorithm:

```
Step 1: Start
Step 2: Declare the variables i,j,l,m,n and a. Set the value of i and a to 1.
Step 3: Input the number of rows(n).
Step 4: Set the value of m to n.(m=n)
Step 5: Start the while loop checking whether i is less than or equal to n. And do the following
        operations:
            • Set the value of l and j to 1.
            • [ Go To Step 6 ]
            • [ Go To Step 7 ]
            • Print “\n” (Break or new line)
            • i++, m-- and a+=2 or (a=a+2)
            • [ Go To Step 8 ]
Step 6: Start the while loop checking whether l is less than m. And do the following operations:
            • Print “\t” (tab or space)
            • l++ or l+1
            • [ Back To Step 5 ]
Step 7: Start the while loop checking whether j is less than or equal to a. And do the following
        operations:
            • Print “*”.
            • j++ or j+1
            • [ Back To Step 5 ]
Step 8: End
```

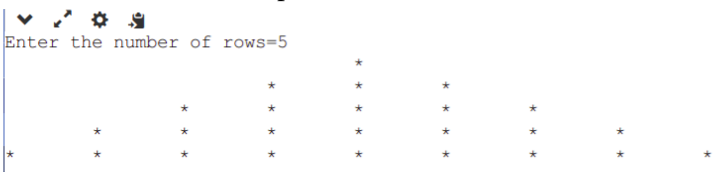
#Program:



C yugin22.c > ...

```c
#include <stdio.h>
int main()
{
    int i,j,l,m,n,a;
    printf("Enter the number of rows=");
    scanf("%d",&n);
    i=a=1;
    m=n;
    while(i<=n)
    {
        l=1;
        while (l<m)
        {
            printf("\t");
            l++;
        }
        j=1;
        while (j<=a)
        {
            printf("* \t");
            j++;
        }
        printf("\n");
        i++;
        m--;
        a+=2;
    }
    return 0;
}
```

#Output:



```
Enter the number of rows=5
                                *
                        *       *       *
                *       *       *       *       *
        *       *       *       *       *       *       *
*       *       *       *       *       *       *       *       *
```

#Flowchart of 21:

#Flowchart of 22:

*** * ***