

AMCAT Coding Question -1 :

Please comment down the code in other languages as well below –

C Program to check if two given matrices are identical

```
#include <stdio.h>
#define N 4

// This function returns 1 if A[][] and B[][] are identical
// otherwise returns 0
int areSame(int A[][N], int B[][N])
{
    int i, j;
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            if (A[i][j] != B[i][j])
                return 0;
    return 1;
}

int main()
{
    int A[N][N] = { {1, 1, 1, 1},
                    {2, 2, 2, 2},
                    {3, 3, 3, 3},
                    {4, 4, 4, 4}};

    int B[N][N] = { {1, 1, 1, 1},
                    {2, 2, 2, 2},
                    {3, 3, 3, 3},
                    {4, 4, 4, 4}};

    if (areSame(A, B))
        printf("Matrices are identical");
    else
        printf("Matrices are not identical");
    return 0;
}
```

Ques. Print the following Pattern and get the output to match test cases?

To print the pattern like
for n=3
the program should print
1 1 1 2
3 2 2 2
3 3 3 4

Program in C++

```
#include <iostream>

using namespace std;

int main()
{
    int n=3,c=n-1;
    for(int i=1;i<=n;i++)
    {
        if(i%2==0)
            cout<<c++;
        for(int j=1;j<=n;j++)
        {
            cout<<i;
        }
        if(i%2!=0)
            cout<<c++;
        cout<<"\n";
    }

    return 0;
}
```

Program in C

```
#include
int main(void) {
    int i,j,n=3,c=n-1;
    for(i=1;i<=n;i++)
    {
        if(i%2==0)
            printf("%d",c++);
        for(j=1;j<=n;j++)
        {
```

```

printf("%d",i);
}
if(i%2!=0)
printf("%d",c++);
printf("\n");
}

return 0;
}

```

Code in Java

```

public class Practice{
public static void main(String[] args){
PrintPat(3); }
public static void PrintPat(int a)
{ int n=1,i,j=1;
while(n<=a){
if(n%2!=0){
for(i=1;i<=a;i++)
System.out.print(n);

System.out.print(++j);
System.out.println();
}
else{
System.out.print(++j);
for(i=1;i<=a;i++)
System.out.print(n);
System.out.println();
}
n++;
}}}

```

Trapezium Pattern in C, Java

Here you will learn how to print the Trapezium Pattern in C language and Trapezium Pattern program in Java, Trapezium Pattern print in C++ –

Table of Contents

1. [Trapezium Pattern in C](#)
2. [Trapezium Pattern in C++](#)
3. [Trapezium Pattern in Java](#)

Ques. To print the trapezium pattern?

Please also post your code in the comments in different languages or same languages with short or better time complexity code.

```
If N = 4
1*2*3*4*17*18*19*20
  5*6*7*14*15*16
    8*9*12*13
      10*11
If n = 5
1*2*3*4*5*26*27*28*29*30
  6*7*8*9*22*23*24*25
    10*11*12*19*20*21
      13*14*17*18
        15*16
If N = 2
1*2*5*6
  3*4
```

In C

```
#include
int main(){
int n=5,num=1,i=1,space=0,k=1,number=n;
for(i=0;i<n;i++)
{
for(int j=1;j<=space;j++)
{

printf("-");

}
for(int m=1;m<2*n-space;m++)
{
if(m%2==0)
printf("%s","*");
else
printf("%d",num++);
}
printf("%s","*");
for(int l=1;l<2*n-space;l++)
{
if(l%2==0)
```

```

printf("%s","*");
else
{
printf("%d",k+number*number);
k++;
}
}
number--;

space=space+2;
printf("\n");
}
return 0;
}

```

In C++

```

#include
using namespace std;
int main(){
int n=4,num=1,i=1,space=0,k=1,number=n;
for(i=0;i<n;i++)
{
for(int j=1;j<=space;j++)
{

cout<<"- ";

}
for(int m=1;m<2*n-space;m++)
{
if(m%2==0)
cout<<"*";
else
cout<<num++;
}
cout<<"*";
for(int l=1;l<2*n-space;l++)
{
if(l%2==0)
cout<<"*";
else
{
cout<<k+number*number;
k++;
}
}
}

```

```

}
number--;

space=space+2;
cout<<endl;
}
return 0;
}

```

Trapezium Pattern program in Java

```

public class Pattern {

    public static void main(String[] args) {

        int count1=0,count2=0;
        int N=4;
        for(int i=N;i>=1;i--) {
            for(int j=N;j>i;j--) System.out.print(" ");

            for(int k=1;k<=i;k++) System.out.print(++count1+"*");

            for(int l=1;l<=i;l++) {
                System.out.print(++count2+i*i);
                if(l!=i) System.out.print("*");
            }
            System.out.println();
        }

    }
}

```

Ques. Programming Pattern to Print 2*N Number of rows for input Pattern?

```

3
44
555
6666
555
44
3

```

Code in C++

C/C++ Program to Print- 3 44 555 6666 6666 555 44 3

```

#include <iostream>
using namespace std;
int main()
{
    int n=4,num=n-1;
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=i;j++)
            cout<<num;
        num++;
        cout<<endl; } num--; for(int i=n;i>=1;i--)
    {
        for(int j=1;j<=i;j++)
            cout<<num;
        num--;
        cout<<endl;
    }
    return 0;
}

```

Please do comment the code in other languages :).

Code in Java –

Java Program to Print- 3 44 555 6666 6666 555 44 3

```

public class Pattern {

    public static void main(String[] args) {

        int N=4;
        for(int i=1;i<=N;i++) {
            for(int j=1;j<=i;j++) System.out.print(i+2); System.out.println(); } for(int
i=N-1;i>=1;i--) {
            for(int k=1;k<=i;k++) System.out.print(i+2);

            System.out.println();
        }
    }
}

```

Ques. Print the following pattern-

C, Java, C++ Program to print 1*2*3*10*11*12 4*5*8*9 6*7 Pattern

```

1*2*3*10*11*12
  4*5*8*9
    6*7

```

Problem: To find GCD of two number.

Solution:

```
#include <iostream>

using namespace std;
int gcd_iter(int u, int v)
{
    int t;
    while (v)
    {
        t = u;
        u = v;
        v = t % v;
    }
    return u < 0 ? -u : u;
}
int main()
{
    int n=3,m=6;
    int result=gcd_iter(n,m);
    cout<<result;
    return 0;
}
```

PRINTING PATTERN:

PREREQUISITE:

Basic knowledge of C language and loops.

ALGORITHM:

1. Take number of rows as input from the user (length of side of the square) and store it in any variable ('l' in this case).
2. Run a loop 'l' number of times to iterate through the rows . From $i=0$ to $i<l$. The loop should be structured as *for*($i=0;i<l;i++$) .
3. Run a nested loop inside the previous loop to iterate through the columns. From $j=0$ to $j<l$. The loop should be structured as *for*($j=0;j<l;j++$) .
4. Print '*' inside the nested loop to print '*'s in all the columns of a row.
5. Move to the next line by printing a new line. *printf*("\\n") .

Code in C:

```
#include<stdio.h>
int main()
{
    int i,j,l;    //declaring integers i,j for loops and l for number of rows
    printf("Enter the number of rows/columns\\n"); //Asking user for input
    scanf("%d",&l);    //Taking the input for number of rows
    for(int i=0;i<l;i++) //Outer loop for number of rows
    {
        for(int j=0;j<l;j++) //Inner loop for number of columns in each row
        {
            printf("*");    //Printing '*' in each column of a row.
        }
        printf("\\n");    //Printing a new line after each row has been printed.
    }
}
```

PRINTING PATTERN:

* *

* *

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take number of rows/columns as input from the user (length of side of square) and store it in any variable ('l' in this case).
2. Run a loop 'l' number of times to iterate through all the rows. From $i=0$ to $i<l$. The loop should be structured as *for*($i=0$; $i<l$; $i++$).
3. Run a nested loop inside the main loop for printing stars . From $j=0$ to $j<l$. The loop should be structured as *for*($j=0$; $j<l$: $j++$).
4. Inside the above loop print stars only *if* $i=0$ or $i=l-1$ or $j=0$ or $j=l-1$ in all other cases print a blank space.
5. Move to the next line by printing a new line. *printf*("\\n").

Code in C:

```
#include<stdio.h>
int main()
{
    int i,j,l;    //declaring integers i,j for loops and l for the number of rows
    printf(" Enter the number of rows\\n");    //Asking user for input
    scanf("%d",&l);    //taking input for number of rows and saving in variable l
    for(i=0;i<l;i++) //Outer loop for number of rows
    {
        for(j=0;j<l;j++) //Inner loop for printing stars in each column of a
row
        {
            if(i==0 || i==l-1 || j==0 || j==l-1) // condition for printing
stars
            {
                printf("*");    // printing stars
            }
            else    // else condition to print spaces
            {
                printf(" ");    //printing spaces
            }
        }
        printf("\\n");    //Printing a new line after a row has been printed
    }
}
```

PRINTING PATTERN:

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM

1. Take the number of rows as input from the user (length of side of rhombus) and store it in any variable.('l' in this case).
2. Run a loop 'l' number of times to iterate through each of the rows. From $i=0$ to $i<l$. The loop should be structured as *for*($i=0$; $i<l$; $i++$).
3. Run a nested loop inside the main loop to print the spaces before the rhombus. From $j=0$ to $j<i$. The loop should be structured as *for*($j=0$; $j<i$; $j++$).
4. Run another nested loop inside the main loop after the previous loop to print the stars in each column of a row. From $j=0$ to $j<l$. The loop should be structured as *for*($j=0$; $j<l$; $j++$).
5. Move to the next line by printing a new line . *printf*("n").

Code in C:

```
#include<stdio.h>
int main()
{
int i,j,l; //declaring integer variables i,j for loops and l for number of
rows/columns
printf("enter number of rows/columns\n"); //asking the user for input
scanf("%d",&l); //taking input from the user
for(i=0;i<l;i++) //loop controlling number of rows
{
    for(j=0;j<i;j++) //inner loop for spaces
    {
        printf(" ");
    }
    for(j=0;j<l;j++) //inner loop for printing the stars in each column of
a row
    {
        printf("*");
    }
    printf("\n"); // printing a new line after each row
}
}
```

PRINTING PATTERN:

```
*  
  
***  
  
*****  
  
*****
```

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$; $i++$).
3. Run a nested loop inside the main loop to print the spaces before the pyramid. From $k=r$ to $k>i+1$. The loop should be structured as *for*($k=r$; $k>i+1$; $k--$).
4. Inside this nested loop print white space.
5. Run another nested loop inside the main loop after the previous loop to print the stars in each column of a row. From $j=0$ to $j<=i*2$. The loop should be structured as *for*($j=0$; $j<=i*2$; $j++$).
6. Inside this nested loop print star.
7. Move to the next line by printing a new line . *printf*("n").

Code in C:

```
#include<stdio.h>  
int main()  
{  
    int i,j,k,r;        //declaring integer variables i,j,k for loops and r for  
    number of rows  
    printf("Enter the number of rows :n");        //Asking user for input  
    scanf("%d",&r);        //saving number of rows in variable r  
    for(i=0;i<r;i++)        //outer loop for number of rows  
    {  
        for(k=r;k>i+1;k--)        //nested loop for number of spaces  
        {  
            printf(" ");        //printing spaces  
        }  
        for(j=0;j<=i*2;j++)        //nested loop for printing stars  
        {  
            printf("*");        //printing stars  
        }  
        printf("\n");        //printing newline  
    }  
}
```

PRINTING PATTERN:

```

    *

  * *

 *  *

*****
```

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$; $i++$).
3. Run a nested loop inside the main loop to print the spaces before the pyramid. From $k=r$ to $k>i+1$. The loop should be structured as *for*($k=r$; $k>i+1$; $k--$).
4. Inside this nested loop print white space.
5. Run another nested loop inside the main loop after the previous loop to print the stars in each column of a row. From $j=0$ to $j<=i*2$. The loop should be structured as *for*($j=0$; $j<=i*2$; $j++$).
6. Inside this nested loop print star under the condition *if*($i==r-1$).
7. Under else use a nested if else statement to print the remaining stars *if*($j==0$ || $j>=i*2$) and else print *white space*
8. Inside main loop move to the next line by printing a new line . *printf*("\\n").

Code in C:

```
#include<stdio.h>
int main()
{
    int i,j,k,r;      //declaring integer variables i,j,k for loops and r for
    number of rows
    printf("Enter the number of rows :\\n");      //Asking user for input
    scanf("%d",&r);      //saving number of rows in variable r
    for(i=0;i<r;i++)      //outer loop for number of rows
    {
        for(k=r;k>i+1;k--)//nested loop for number of spaces
        {
            printf(" ");//printing spaces
        }
        for(j=0;j<=i*2;j++)//nested loop for printing stars
```

```

    {
        if(i==r-1) //condition to print last row
        {
            printf("*"); //printing stars in last row
        }
        else //else condition to print rest of the stars
        {
            if(j==0||j>=i*2) //condition to print stars
            {
                printf("*"); //printing stars
            }
            else //else condition to print spaces
            {
                printf(" "); //printing spaces
            }
        }
    }
    printf("\n"); //printing newline
}
}

```

PRINTING PATTERN:

*

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=r$ to $i>0$. The loop should be structured as *for*($i=r$; $i>0$: $i--$).
3. Run a nested loop inside the main loop to print the spaces before the pyramid. From $k=r$ to $k>i+1$. The loop should be structured as *for*($k=r$; $k>i+1$; $k--$).
4. Inside this loop print white space.
5. Run another nested loop after the previous loop to print the stars in each column of a row. From $j=0$ to $j<i*2-1$. The loop should be structured as *for*($j=0$; $j<i*2$; $j++$).
6. Inside this loop print star.

7. Move to the next line by printing a new line . `printf("\n")`

CODE IN C:

```
#include<stdio.h>
int main()
{
int i,j,k,r;      //declaring integer variables i,j,k for loops and r for
number of rows
printf("Enter the number of rows :\n");      //Asking user for input
scanf("%d",&r);      //saving number of rows in variable r
for(i=r;i>0;i--)      //outer loop for number of rows
{
    for(k=r;k>i;k--)      //nested loop for spaces before the pyramid
    {
        printf(" ");      //printing white space
    }
    for(j=0;j<i*2-1;j++)      //loop for printing stars
    {
        printf("*");      //printing stars
    }
    printf("\n");      //printing new line after each row
}
}
```

PRINTING PATTERN:

* *

* *

*

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable.('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=r$ to $i>0$. The loop should be structured as `for(i=r ; i>0 : i--)`.

3. Run a nested loop inside the main loop to print the spaces before the pyramid. From $k=r$ to $k>i+1$. The loop should be structured as *for(k=r; k>i+1 ;k-)*.
4. Inside this nested loop print white space.
5. Run another nested loop inside the main loop after the previous loop to print the stars in each column of a row. From $j=0$ to $j<i*2-1$. The loop should be structured as *for(j=0 ; j<i*2-1; j++)*.
6. Print star under the *if condition (i==r)*.
7. Under the else condition use another if condition to print the rest of the stars.
8. Use *if(j==0 || j==i*2-2)* to print stars.
9. Else print white space
10. In the main loop print a new line to move to the next line.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,k,r;    //declaring integer variables i,j,k for loops and r for
    number of rows
    printf("Enter the number of rows :\n");    //Asking user for input
    scanf("%d",&r);    //saving number of rows in variable r
    for(i=r;i>0;i--)    //outer loop for number of rows
    {
        for(k=r;k>i;k--)    //nested loop for spaces before the pyramid
        {
            printf(" ");    //printing white space
        }
        for(j=0;j<i*2-1;j++)    //loop for printing stars
        {
            if(i==r)    //condition to print the first row
            {
                printf("*");    //printing stars in the first row
            }
            else    //else condition for printing the rest of the pyramid
            {
                if(j==0||j==i*2-2)    //if condition to print the starting
                and ending stars in a row
                {
                    printf("*");    //printing stars
                }
                else    //else condition for printing white space
                {
                    printf(" ");    //printing white space
                }
            }
        }
        printf("\n");    //printing newline after each row
    }
}
```


PRINTING PATTERN:

* *

* *

*

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=r$ to $i>0$. The loop should be structured as *for*($i=r$; $i>0$: $i--$).
3. Run a nested loop inside the main loop to print the spaces before the pyramid. From $k=r$ to $k>i$ +1. The loop should be structured as *for*($k=r$; $k>i+1$; $k--$).
4. Inside this nested loop print white space.
5. Run another nested loop inside the main loop after the previous loop to print the stars in each column of a row. From $j=0$ to $j<i*2-1$. The loop should be structured as *for*($j=0$; $j<i*2-1$; $j++$).
6. Print star under the *if condition* ($i==r$).
7. Under the else condition use another if condition to print the rest of the stars.
8. Use *if*($j==0$ || $j==i*2-2$) to print stars.
9. Else print white space
10. In the main loop print a new line to move to the next line.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,k,r;    //declaring integer variables i,j,k for loops and r for
    number of rows
    printf("Enter the number of rows :\n");    //Asking user for input
    scanf("%d",&r);    //saving number of rows in variable r
    for(i=r;i>0;i--)    //outer loop for number of rows
    {
        for(k=r;k>i;k--)    //nested loop for spaces before the pyramid
        {
            printf(" ");    //printing white space
        }
        for(j=0;j<i*2-1;j++)    //loop for printing stars
        {
            if(i==r)    //condition to print the first row
            {
```

```

        printf("*");        //printing stars in the first row
    }
    else        //else condition for printing the rest of the pyramid
    {
        if(j==0 || j==i*2-2)        //if condition to print the starting
and ending stars in a row
        {
            printf("*");        //printing stars
        }
        else        //else condition for printing white space
        {
            printf(" ");        //printing white space
        }
    }
}
printf("\n");        //printing newline after each row
}
}

```

PRINTING PATTERN:

```

*

**

***

****

***

**

*

```

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$; $i++$).
3. Use an if condition to to print the top half of the pyramid. *if* ($i<=r/2$). Then run a loop from $j=0$ to $j<=i$. The loop should be structured as *for*($j=0$; $j<=i$; $j++$)
4. Inside this loop print star.

5. Else run a different loop from $j=i$ to $j<r$. The loop should be structured as *for*($j=i; j<r; j++$).
6. Inside this loop print star.
7. Inside the main loop print a newline to move to the next line after each row is printed.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r;                //declaring integer variables i,j for loops
    and r for number of rows
    printf("Enter the number of rows(odd) :\n"); //Asking user for input
    scanf("%d",&r);           //taking number of rows and saving it in
    variable r
    for(i=0;i<r;i++)          // loop for number of rows
    {
        if(i<=(r/2))          //if condition to print the top half
        {
            for(j=0;j<=i;j++)  // loop for stars per each row
            {
                printf("*");    //printing stars
            }
        }
        else                    //else condition to print the bottom half
        {
            for(j=i;j<r;j++)    //loop for printing
            {
                printf("*");    //printing stars
            }
        }

        printf("\n");          // printing newline after each row
    }
}
```

PRINTING PATTERN:

```

    *

   **

  ***

 ****

 ***

```

**

*

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$; $i++$).
3. Use an if condition to to print the top half of the diamond. *if* ($i\leq r/2$). Then run a loop from $k=r/2$ to $k>i$. The loop should be structured as *for*($k=r/2$; $k>i$; $k--$). Inside this loop print space.
4. After this inside the if block run another loop from $j=0$ to $j\leq i$. The loop should be structured as *for*($j=0$; $j\leq i$; $j++$)
5. Inside this loop print star.
6. Else run a different loop from $k=r/2$ to $k<i$, The loop should be structured as *for*($k=r/2$; $k<i$; $k++$). Inside this loop print space
7. Run another loop inside the else block from $j=i$ to $j<r$. The loop should be structured as *for*($j=i$; $j<r$; $j++$).
8. Inside this loop print star.
9. Inside the main loop print a newline to move to the next line after each row is printed.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,k,r;           //declaring integer variables i,j,k for loops and r
    for number of rows
    printf("Enter the number of rows(odd) :\n");    //Asking user for input
    scanf("%d",&r);           //taking number of rows and saving it in
    variable r
    for(i=0;i<r;i++)         // loop for number of rows
    {
        if(i<=(r/2))        //if condition to print the top half
        {
            for(k=r/2;k>i;k--)    //loop to print spaces before the top
half of diamond
            {
                printf(" ");    //printing spaces
            }
            for(j=0;j<=i;j++)    // loop for stars per each row
            {
                printf("*");    //printing stars
            }
        }
        else                //else condition for printing the lower half
of the diamond
    }
```

```

        {
            for(k=r/2;k<i;k++)        //loop for printing spaces before the
lower half
                {
                    printf(" ");        //printing spaces
                }
            for(j=i;j<r;j++)        //loop for printing stars
                {
                    printf("*");        //printing stars
                }

            }

        printf("\n");        // printing newline after each row
    }
}

```

PRINTING PATTERN:

```

    *

 ***

*****

*****

*****

 ***

 *

```

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for* ($i=0$; $i<r$: $i++$).
3. Use an if condition to to print the top half of the diamond. *if* ($i\leq r/2$). Then run a loop from $k=r/2$ to $k>i$. The loop should be structured as *for*($k=r/2$; $k>i$; $k--$). Inside this loop print space.

4. After this inside the if block run another loop from $j=0$ to $j \leq i*2$. The loop should be structured as *for*($j=0; j \leq i*2; j++$)
5. Inside this loop print star.
6. Else run a different loop from $k=r/2$ to $k < i$, The loop should be structured as *for*($k=r/2; k < i; k++$). Inside this loop print space
7. Run another loop inside the else block from $j=i$ to $j < ((r-i)*2)-1$. The loop should be structured as *for*($j=0; j < ((r-i)*2)-1; j++$).
8. Inside this loop print star.
9. Inside the main loop print a newline to move to the next line after each row is printed.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,k,r;          //declaring integer variables i,j,k for loops and r for
    number of rows
    printf("Enter the number of rows :\n"); //Asking user for input
    scanf("%d",&r);        //saving number of rows in variable r
    for(i=0;i<r;i++)       //outer loop for number of rows
    {
        if(i<=r/2)         //if condition to print the top half
        of diamond
        {
            for(k=r/2;k>i;k--) //nested loop for number of spaces
            {
                printf(" "); //printing spaces
            }
            for(j=0;j<=i*2;j++) //nested loop for printing stars
            {
                printf("*"); //printing stars
            }
        }
        else               //else condition to print the bottom
        half of the diamond
        {
            for(k=r/2;k<i;k++) //nested loop to print spaces
            {
                printf(" "); //printing spaces
            }
            for(j=0;j<((r-i)*2)-1;j++) //loop to print star
            {
                printf("*"); //printing stars
            }
        }
        printf("\n"); //printing newline
    }
}
```

PRINTING PATTERN:

333

313

323

333

PREREQUISITE:

Basic knowledge of C language and loops.

ALGORITHM:

1. Take number of rows as input from the user and save it in any variable ('r' in this case).
2. Run a loop 'r' number of times to iterate through the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$; $i++$).
3. Inside this if condition use an id condition to print the top and bottom rows. *if*($i==0$ || $i==r-1$).
4. Inside this run a nested loop to iterate through the columns. From $j=0$ to $j<3$. The loop should be structured as *for*($j=0$; $j<3$; $j++$).
5. Print "3" in this loop.
6. Write an else condition. Inside it a run a similar loop . From $j=0$ to $j<3$. The loop should be structured as *for*($j=0$; $j<3$; $j++$).
7. Use an if condition. *if*($j==0$ || $j==2$). And print "3".
8. Else print the iterating variable for rows.
9. Inside the main loop print a newline

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r;                                     //declaring integer
    variables i,j for loops , r for number of rows
    printf("Enter the number of rows :\n");        //asking user for input
    scanf("%d",&r);                                //saving number of rows
    in variable r
    for(i=0;i<r;i++)                               //loop for number of rows
    {
        if(i==0 || i==r-1)                         //if condition to print
        first and last row
        {
            for(j=0;j<3;j++)                        //loop to print first and
            last row
            {
                printf("3");                        //printing 3
            }
        }
    }
}
```

```

        }
        else //else to print rest of
the square
        {
            for(j=0;j<3;j++) //loop to print the rest
of the square
            {
                if(j==0 || j==2) //if condition to print
the outer values
                {
                    printf("3"); //printing 3
                }
                else//else condition to
                {
                    printf("%d",i); //Printing the middle
column
                }
            }
        }
        printf("\n"); //printing newline
    }
}

```

PRINTING PATTERN:

1

23

456

78910

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Initialise an integer variable count=0 for incrementing
2. Take the number of rows as input from the user and store it in any variable.('r' in this case).
3. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for(i=0 ; i<r : i++)*.

4. Run a nested loop inside the main loop to print the digits in each row of the triangle.
From $j=0$ to $j<i$. The loop should be structured as *for*($j=0; j<i; j++$).
5. In the nested loop increment count and print it.
6. In the main loop print a new line.

Code in C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;                //declaring integer variables i,j
    for loops , r for number of rows and count for increment in value
    count=0;                        //initialising count
    printf("Enter the number of rows :\n"); //Asking user for input
    scanf("%d",&r);                  //taking number of rows and saving it
    in variable r
    for(i=0;i<r;i++)                // loop for number of rows
    {
        for(j=0;j<=i;j++)           // loop for digits per each row
        {
            count++;                //incrementing count
            printf("%d",count);     //printing digits
        }
        printf("\n");               // printing newline after each row
    }
}
```

PRINTING PATTERN:

10987

654

32

1

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Initialise an integer variable count=0 for finding the maximum number.
2. Take the number of rows as input from the user and store it in any variable.(‘r’ in this case).

3. Run a loop 'r' number of times to iterate through each of the rows. From $i=r$ to $i>0$. The loop should be structured as *for*($i=r$; $i>0$: $i--$)
4. Inside this loop calculate the maximum value of count by $count+=i$.
5. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$: $i++$).
6. Run a nested loop inside the main loop to print the digits in each row of the triangle. From $j=r$ to $j>i$. The loop should be structured as *for*($j=r$; $j>i$; $j--$).
7. In the nested loop print count and then decrement it.
8. In the main loop print a new line.

Code in C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;                //declaring integer variables i,j
    for loops , r for number of rows, count for decrement in value
    count=0;                        //initialising count
    printf("Enter the number of rows :\n"); //Asking user for input
    scanf("%d",&r);                //taking number of rows and saving it
    in variable r
    for(i=r;i>0;i--)                //loop to find the maximum value to
    print
    {
        count+=i;                  //incrementing count
    }
    for(i=0;i<r;i++)                // loop for number of rows
    {
        for(j=r;j>i;j--)           // loop for digits per each row
        {
            printf("%d",count);    //printing digit
            count--;               //decrementing count
        }
        printf("\n");              // printing newline after each row
    }
}
```

PRINTING PATTERN:

6666

555

44

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Initialise an integer variable $\text{count} = r + 2$.
3. Run a loop 'r' number of times to iterate through each of the rows. From $i = 0$ to $i < r$. The loop should be structured as *for*($i = 0$; $i < r$; $i++$).
4. Run a nested loop inside the main loop to print the digits in each row of the triangle. From $j = r$ to $j > i$. The loop should be structured as *for*($j = r$; $j > i$; $j--$).
5. In the nested loop print count.
6. In the main loop decrement count and print a new line.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;//declaring integer variables i,j for loops , r for number of
    rows
    printf("Enter the number of rows/columns :\n");//asking user for the number
    of rows;
    scanf("%d",&r);//taking number of rows and saving in variable r
    count=r+2;
    for(i=0;i<r;i++) //loop for number of rows
    {
        for(j=r;j>i;j--)//loop to print digit in every column of a row
        {
            printf("%d",count);//printing digit
        }
        count--;
        printf("\n");//printing newline
    }
}
```

PRINTING PATTERN:

3

44

555

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Initialise an integer variable count=3 for incrementing
2. Take the number of rows as input from the user and store it in any variable.('r' in this case).
3. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$; $i++$).
4. Run a nested loop inside the main loop to print the digits in each row of the triangle. From $j=0$ to $j<i$. The loop should be structured as *for*($j=0$; $j<i$; $j++$).
5. In the nested loop print count.
6. In the main loop increment count and print a new line.

Code in C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;                                //declaring integer
    variables i,j for loops , r for number of rows
    printf("Enter the number of rows/columns :\n"); //asking user for the
    number of rows;
    scanf("%d",&r);                                  //taking number of rows
    and saving in variable r
    count=3;                                         //intialising count =3
    for(i=0;i<r;i++)                                //loop for number of rows
    {
        for(j=0;j<=i;j++)                            //loop to print digit in
        every column of a row
        {
            printf("%d",count);                       //printing digit
        }
        count++;                                     //incrementing count
        printf("\n");                                  //printing newline
    }
}
```

PRINTING PATTERN:

3

45

678

9101112

678

45

3

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$: $i++$).
3. Use an if condition to to print the top half of the pyramid. *if* ($i\leq r/2$). Then initialise $count1=count+1$ and Then run a loop from $j=0$ to $j\leq i$. The loop should be structured as *for*($j=0$; $j\leq i$; $j++$)
4. Inside this loop increment count and then print it.
5. Outside this loop print a newline.
6. Else $count=count1-(r-i)$; and $count1=count$ (to print the values accordingly).
7. Run a different loop from $j=i$ to $j<r$. The loop should be structured as *for*($j=i$; $j<r$; $j++$).
8. Inside this loop print count and then increment it.
9. Inside the main loop print a newline to move to the next line after each row is printed.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count,count1;           //declaring integer variables i,j for
    loops , r for number of rows and count and count1 for increment in value
    count=2; //initialising count
    printf("Enter the number of rows :\n"); //Asking user for input
    scanf("%d",&r);                   //taking number of rows and saving it
    in variable r
    for(i=0;i<r;i++)                 // loop for number of rows
    {
        if(i<=r/2)                   //if statement to print top half
        {
            count1=count+1;           //giving the changed value to count1
            for(j=0;j<=i;j++)         // loop for digits per each row
            {
                count++;              //incrementing count
                printf("%d",count);   //printing digits
            }
            printf("\n");             // printing newline after each row
        }
    }
}
```

```

    }
    else //else statement for bottom half
    {
        count=count1-(r-i); //reevaluating count
        count1=count; //giving value of count to count1
        for(j=i; j<r; j++) //loop for bottom half
        {
            printf("%d",count); //printing count
            count++; //incrementing count
        }
        printf("\n"); //printing newline
    }
}
}

```

PRINTING PATTERN:

```

3
54
876
1211109
876
54
3

```

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for* ($i=0$; $i<r$; $i++$).
3. Use an if condition to to print the top half of the pyramid. *if* ($i\leq r/2$). Then initialise count=count1 and Then run a loop from $j=0$ to $j\leq i$. The loop should be structured as *for* ($j=0$; $j\leq i$; $j++$)
4. Inside this loop increment count.

5. Outside this loop initialise `count1=count`.
6. Then run a loop from `j=0` to `j<=i`. The loop should be structured as `for(j=0 ; j<=i ; j++)`
7. Inside this loop decrement count and then print it
8. Outside the loop print a newline
9. Else
10. Run a different loop from `j=i` to `j<r`. The loop should be structured as `for(j=i ; j<r ; j++)`.
11. Inside this loop decrement count and then print it.
12. Inside the main loop print a newline to move to the next line after each row is printed.

CODE IN C:

```
#include<stdio.h>
int main()
{
int i,j,r,count,count1;
count1=3; //declaring integer variables i,j for
loops , r for number of rows and count for increment in value
count=0; //initialising count
printf("Enter the number of rows(odd) :\n"); //Asking user for input
scanf("%d",&r); //taking number of rows and saving it
in variable r
for(i=0;i<r;i++) // loop for number of rows
{
if(i<=r/2) //if condition for top half
{
count=count1; //copying value
for(j=0;j<=i;j++) // loop for digits per each row
{
count++; //incrementing count
}
count1=count; //copying value
for(j=0;j<=i;j++) // loop for digits per each row
{
count--; //incrementing count
printf("%d",count); //printing digits
}

printf("\n"); // printing newline after each row
}
else
{
for(j=i;j<r;j++) //loop for lower half
{
count--; //decrementing count
printf("%d",count); //printing digits
}
printf("\n"); //printing newline
}
}
}
```

PRINTING PATTERN:

2

33

444

5555

5555

444

33

2

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$; $i++$).
3. Use an if condition to print the top half of the pyramid. *if* ($i\leq r/2$) run a loop from $j=0$ to $j\leq i$. The loop should be structured as *for*($j=0$; $j\leq i$; $j++$)
4. Inside this loop print count.
5. Outside this loop increment count and print a newline
6. Else decrement count
7. Run a different loop from $j=0$ to $j<r-i+1$. The loop should be structured as *for*($j=0$; $j<r-i+1$; $j++$).
8. Inside this loop print count.
9. After the loop print a newline.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count; //declaring integer variables i,j for loops , r for number
    of rows
    printf("Enter the number of rows/columns :\n"); //asking user for the number
    of rows;
    scanf("%d",&r); //taking number of rows and saving in variable r
```



```

count=2; //initialising count =3
for(i=1;i<=r;i++) //loop for number of rows
{
    if(i<=r/2)
    {
        for(j=1;j<=i;j++) //loop to print digit in every column of a row
        {
            printf("%d",count); //printing digit
        }
        count++; //incrementing count
        printf("\n"); //printing newline
    }
    else
    {
        count--;
        for(j=0;j<r-i+1;j++)
        {
            printf("%d",count);
        }
        printf("\n");
    }
}
}

```

PRINTING PATTERN:

2

34

567

891011

891011

567

34

2

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$: $i++$).
3. Use an if condition to to print the top half of the pyramid. *if* ($i\leq r/2$). Then initialise $count1=count+1$ and Then run a loop from $j=0$ to $j\leq i$. The loop should be structured as *for*($j=0$; $j\leq i$; $j++$)
4. Inside this loop increment count and print it.
5. Outside this loop print a newline.
6. Else initialise $count=count1$;
7. Run a different loop from $j=i$ to $j<r-i$. The loop should be structured as *for*($j=i$; $j<r-i$; $j++$).
8. Inside this loop print count and increment it
9. outside the loop modify the value of count1 as $count1=count1-(r-i)+1$
10. then print a newline

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count,count1;           //declaring integer variables
    i,j for loops , r for number of rows and count for increment in value
    count=1; //initialising count
    printf("Enter the number of rows(even) :\n"); //Asking user for input
    scanf("%d",&r); //taking number of rows and
    saving it in variable r
    for(i=0;i<r;i++) // loop for number of rows
    {
        if(i<r/2)
        {
            count1=count+1;
            for(j=0;j<=i;j++) // loop for digits per each
row
                {
                    count++; //incrementing count
                    printf("%d",count); //printing digits
                }
            printf("\n"); // printing newline after each
row
        }
        else //else condition for bottom
half
        {
            count=count1; //copying value
            for(j=0;j<r-i;j++) //loop to print digits
            {
                printf("%d",count); //printing digits
            }
        }
    }
}
```

```

        count++;                                //incrementing count
    }
    count1=count1-(r-i)+1;                        //copying value
    printf("\n");                                //printing newline
}
}
}

```

```

2
43
765
1110198
1110198
765
43
2

```

PRINTING PATTERN:

1*2*3*4

5*6*7*8

9*10*11*12

13*14*15*16

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows/columns as input from the user and store it in any variable.('l' in this case).
2. Run a loop 'l' number of times to iterate through each of the rows. From $i=0$ to $i<l$. The loop should be structured as *for*($i=0$; $i<l$; $i++$).
3. Inside this loop run another nested loop to iterate through the columns. From $j=0$ to $j<l$. The loop should be structured as *for*($j=0$; $j<l$; $j++$).
4. increment count and the run an if condition *if*($j==l-1$) .
5. which means if it is the last column then print only count

6. Else print a star after count
7. outside the loop print a newline

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,l,count=0;                                //declaring integers i,j for
    loops and l for number of rows
    printf("Enter the number of rows/columns\n");      //Asking user for input
    scanf("%d",&l);                                     //Taking the input for number
    of rows
    for(int i=0;i<l;i++)                                //Outer loop for number of
    rows
    {
        for(int j=0;j<l;j++)                            //Inner loop for number of
        columns in each row
        {
            count++;                                     //incrementing count
            if(j==l-1)                                   //running if statement to not
            print star after the last column of digits
            {
                printf("%d",count);                     //printing count
            }
            else                                         //else statement to print star
            after count
            {
                printf("%d*",count);                     //printing star after count
            }
        }
        printf("\n");                                   //Printing a new line after
        each row has been printed.
    }
}
```

PRINTING PATTERN:

13*14*15*16

9*10*11*12

5*6*7*8

1*2*3*4

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows/columns as input from the user and store it in any variable.('l' in this case).
2. Run a loop 'l' number of times to iterate through each of the rows. From $i=0$ to $i<l$. The loop should be structured as *for*($i=0 ; i<l ; i++$).
3. Inside this loop run another nested loop to iterate through the columns. From $j=0$ to $j<l$. The loop should be structured as *for*($j=0 ; j<l ; j++$).
4. And increment count to get the max number required.
5. Run a loop 'l' number of times to iterate through each of the rows. From $i=0$ to $i<l$. The loop should be structured as *for*($i=0 ; i<l ; i++$).
6. Inside this loop run another nested loop to iterate through the columns. From $j=0$ to $j<l$. The loop should be structured as *for*($j=0 ; j<l ; j++$).
7. increment count and run an if condition *if*($j==l-1$).
8. if true then print count else print count and a star after it.
9. outside the nested loop reinitialize $count=count-2*l$ then print a newline

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,l,count=0; //declaring integers i,j for loops and l for number of
    rows
    printf("Enter the number of rows/columns\n"); //Asking user for input
    scanf("%d",&l); //Taking the input for number of rows
    for(int i=0;i<l;i++) //Outer loop for number of rows
    {
        for(j=0;j<l-1;j++)
        {
            count++;
        }
    }
    for(i=0;i<l;i++)
    {
        for(j=0;j<l;j++)
        {
            count++;
            if(j==l-1)
            {
                printf("%d",count);
            }
            else
            {
                printf("%d*",count);
            }
        }
        count=count-2*l;
        printf("\n");
    }
}
```

For any input Number N print the following code – for below code N=4.

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

Basic incrementing Squared Number-Star Pattern + Basic incrementing inverted Squared Number-Star Pattern (alternate)

Had N value been 5

Then output –

```
1 * 2 * 3 * 4 * 5
11 * 12 * 13 * 14 * 15
21 * 22 * 23 * 24 * 25
16 * 17 * 18 * 19 * 20
6 * 7 * 8 * 9 * 10
```

Code in C

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Enter the Number of Rows : ";
    cin>>n;
    int p=n;
    if(n>=1 && n<=100)
    {
        for(int i=1;i<=n;i+=2)
        {
            int k=(i-1)*n+1;
            for(int j=0;j<n-1;j++)
            {
                cout<<k<<" * ";
                k++;
            }
            cout<<k<<" ";
            cout<<endl;
        }
        if(n%2!=0)
        {
            p=n-1;
        }
    }
}
```

```

for(int i=p;i>0;i-=2)
{
int k=(i-1)*n+1;
for(int j=0;j<n-1;j++)
{
cout<<k<<" * ";
k++;
}
cout<<k<<" ";
cout<<endl;
}
}
else
{
cout<<"Enter a Valid Input(1-100)!" ;
}
return 0;
}

```

PRINTING PATTERN:

1*2*3*4

9*10*11*12

13*14*15*16

5*6*7*8

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows/columns as input from the user and store it in any variable.('l' in this case).
2. Run a loop 'l' number of times to iterate through each of the rows. From $i=0$ to $i<l$. The loop should be structured as *for* ($i=0$; $i<l$; $i++$).
3. Run an if condition *if* ($i==l$) then increment count by l.
4. Run another if condition after the previous one *if* ($i==l-1$) then count should be equal to l.

5. run a nested loop to iterate through the columns. From $j=0$ to $j<l$. The loop should be structured as *for(j=0; j<l; j++)*.
6. increment count and run an if condition *if(j==l-1)*.
7. if true then print count else print count and a star after it.
8. outside the nested loop reinitialize $\text{count}=\text{count}-2*\text{l}$ then print a newline

CODE IN C:

```
#include<stdio.h>
int main()
{
int i,j,l,count=0; //declaring integers i,j for loops and l for number of
rows
printf("Enter the number of rows/columns\n"); //Asking user for input
scanf("%d",&l); //Taking the input for number of rows
for(int i=0;i<l;i++) //Outer loop for number of rows
{
    if(i==1)
    {
        count+=1;
    }
    if(i==l-1)
    {
        count=1;
    }
    for(int j=0;j<l;j++) //Inner loop for number of columns in each row
    {
        count++;
        if(j==l-1)
        {
            printf("%d",count);
        }
        else
        {
            printf("%d*",count);
        }
    }
    printf("\n"); //Printing a new line after each row has been printed.
}
}
```

```
1*2*3*4
9*10*11*12
17*18*19*20
13*14*15*16
5*6*7*8
4*3*2*1
12*11*10*9
8*7*6*5
16*15*14*13
```


PRINTING PATTERN:

1

2*3

4*5*6

7*8*9*10

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows/columns as input from the user and store it in any variable.('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$ The loop should be structured as *for*($i=0$; $i<r$: $i++$).
3. Run a nested loop to iterate through the columns. From $j=0$ to $j<r$. The loop should be structured as *for*($j=0$; $j<r$; $j++$).
4. increment count and run an if condition *if*($j!=0$).
5. if true then print star and count else print only count.
6. outside the nested loop print a newline

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;                                //declaring integer
    variables i,j for loops , r for number of rows and count for increment in
    value
    count=0;                                         //initialising count
    printf("Enter the number of rows :\n");         //Asking user for input
    scanf("%d",&r);                                 //taking number of rows
    and saving it in variable r
    for(i=0;i<r;i++)                                // loop for number of rows
    {
        for(j=0;j<=i;j++)                            // loop for digits per each row
        {
            count++;                                  //incrementing count
            if(j!=0)                                  //if statement to print star and digit
                printf("%d",count);                  //printing star and digit
        }
    }
}
```

```

        else                                     //else statement to print only digit
        printf("%d",count);                       //printing digits
    }
    printf("\n");                                // printing newline after each row
}
}

```

PRINTING PATTERN:

7*8*9*10

4*5*6

2*3

1

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows/columns as input from the user and store it in any variable.('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$ The loop should be structured as *for(i=0 ; i<r: i++)*.
3. and add the 'i' value to count.
4. Run another loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$ The loop should be structured as *for(i=0 ; i<r: i++)*.
5. Reinitialise the value of count as $count=count-r+1$ then give the same value to count1
6. Run a nested loop to iterate through the columns. From $j=0$ to $j<r$. The loop should be structured as *for(j=0 ; j<r; j++)*.
7. increment count and run an if condition *if(j!=r)*.
8. if true then print star and count else print only count.
9. outside the nested loop print a newline
10. Then give value of count1 to count

CODE IN C:

```

#include<stdio.h>
int main()
{
    int i,j,r,count,count1;                                //declaring integer variables
    i,j for loops , r for number of rows and count for increment in value
}

```

```

count=0; //initialising count
printf("Enter the number of rows :\n"); //Asking user for input
scanf("%d",&r); //taking number of rows and saving it in variable r
for(i=1;i<=r;i++) //loop to calculate max value of count
{
    count+=i; //incrementing count
}
for(i=0;i<r;i++) // loop for number of rows
{
    count=count-r+i; //changing value of count
    count1=count; //giving value of count to count1
    for(j=r;j>i;j--) // loop for digits per each row
    {
        count++; //incrementing count
        if(j!=r) //if condition to print star and digit
            printf("%d",count); //printing star and digit
        else //else condition to print only digit
            printf("%d",count); //printing digits
    }
    printf("\n"); //printing newline
    count =count1; // giving count its original value
}
}

```

PRINTING PATTERN:

1

4*5*6

2*3

7*8*9*10

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable.('l' in this case).
2. Run a loop 'l' number of times to iterate through each of the rows. From $i=0$ to $i<l$: The loop should be structured as *for(i=0 ; i<l: i++)*.
3. Run a nested loop to iterate through the columns. From $j=0$ to $j<l$. The loop should be structured as *for(j=0 ; j<l; j++)*.
4. increment count and run an if condition *if(j!=0)*.
5. if true then print star and count else print only count.

6. outside the nested loop print a newline
7. Run another nested loop under the main loop . From $j=0$ to $j \leq i+2$. The loop should be structured as *for*($j=0$; $j \leq i+2$; $j++$).
8. increment count1 and run an if condition *if*($j \neq 0$).
9. if true then print star and count1 else print only count1.
10. outside the nested loop print a newline

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,l,count=0,count1=3; //declaring integers i,j for loops and l for
    number of rows
    printf("Enter the number of rows\n"); //Asking user for input
    scanf("%d",&l); //Taking the input for number of rows
    for(int i=0;i<l/2;i++) //Outer loop for number of rows
    {
        for(j=0;j<=i;j++)
        {
            count++;
            if(j!=0)
            {
                printf("%d",count);
            }
            else
            {
                printf("%d",count);
            }
        }
        printf("\n");
        for(j=0;j<=i+2;j++)
        {
            count1++;
            if(j!=0)
            {
                printf("%d",count1);
            }
            else
            {
                printf("%d",count1);
            }
        }
        printf("\n");
    }
}
```

PRINTING PATTERN:

$2*2$

3*3*3

4*4*4*4

$$3*3*3$$
 $2*2$

1

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as `for(i=0 ; i<r : i++)`.
3. Use an if condition to print the top half of the triangle. `if (i<=r/2)`. Then run a loop from $j=0$ to $j<=i$. The loop should be structured as `for(j=0 ; j<=i ; j++)`
4. Run a nested if statement `if(j!=0)` then print star and $i+1$.
5. Else print only $i+1$.
6. Else statement for the outer if statement: run a loop from $j=i$ to $j<r$. The loop should be structured as `for(j=i ; j<r; j++)`
7. Inside this loop run an if statement `if(j!=i)` then print star and $r-i$.
8. Else just print $r-i$.
9. Inside the main loop print a newline to move to the next line after each row is printed.

CODE IN C:

```
#include<stdio.h>
int main()
{
int i,j,r; //declaring integer variables i,j for loops and r for number of rows
printf("Enter the number of rows(odd) :\n"); //Asking user for input
scanf("%d",&r); //taking number of rows and saving it in variable r
for(i=0;i<r;i++) // loop for number of rows
{
    if(i<=(r/2)) //if condition to print the top half
    {
        for(j=0;j<=i;j++) // loop for stars per each row
        {
            if(j!=0)
            {
                printf("%*d",i+1); //printing stars
```

```

        }
        else
        {
            printf("%d",i+1); //printing stars
        }
    }
}
else //else condition to print the bottom half
{
    for(j=i;j<r;j++) //loop for printing
    {
        if(j!=i)
        {
            printf("%d",r-i); //printing stars
        }
        else
        {
            printf("%d",r-i); //printing stars
        }
    }
}

printf("\n"); // printing newline after each row
}
}

```

PRINTING PATTERN:

1

2*2

3*3*3

4*4*4*4

4*4*4*4

3*3*3

2*2

1

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=1$; $i\leq r$: $i++$).
3. Use an if condition to to print the top half of the pyramid. *if* ($i\leq r/2$) run a loop from $j=0$ to $j\leq i$. The loop should be structured as *for*($j=1$; $j\leq i$; $j++$)
4. Run an if statement *if*($j!=1$). If true the print star and count else only print count.
5. Then in the outer if statement increment count. Then print a newline
6. Inside this loop print count.
7. Outside this loop increment count and print a newline
8. Else decrement count and run a loop from $j=0$ to $j<r-i$. The loop should be structured as *for*($j=0$; $j<r-i$; $j++$). inside the loop run an if statement *if*($j!=0$) then print star and count else just print count.
9. After the loop print a newline.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;                                     //declaring integer
    variables i,j for loops , r for number of rows      //asking user for the
    printf("Enter the number of rows/columns :\n");
    number of rows;
    scanf("%d",&r);                                     //taking number of
    rows and saving in variable r
    count=1;                                             //intialising count =3
    for(i=1;i<=r;i++)                                   //loop for number of
    rows
    {
        if(i<=r/2)
        {
            for(j=1;j<=i;j++)                           //loop to print digit
            in every column of a row
            {
                if(j!=1)
                {
                    printf("%d",count);                  //printing digit
                }
                else
                {
                    printf("%d",count);                  //printing digit
                }
            }
            count++;                                     //incrementing count
            printf("\n");                                 //printing newline
        }
        else
        {
            count--;
```

```

        for(j=0;j<r-i+1;j++)
        {
            if(j!=0)
            {
                printf("%d",count);
            }
            else
            {
                printf("%d",count);
            }
        }
        printf("\n");
    }
}

```

PRINTING PATTERN:

4*4*4*4

3*3*3

2*2

1

1

2*2

3*3*3

4*4*4*4

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable.('r' in this case).
2. Divide the value of 'r' by 2 and replace it in r. And give this value to count
3. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for(i=0 ; i<r : i++)*.

4. Run a loop from $j=r$ to $j>i$. The loop should be structured as *for*($j=r; j>i; j--$)
5. Run an if statement *if*($j!=r$). If true the print star and count else only print count.
6. Then in the outer if statement decrement count. Then print a newline
7. Outside this loop increment count and run a loop from $i=0$ to $i<r$. The loop should be structured as *for*($i=0; i<r; i++$).
8. Run a nested loop from $j=0$ to $j<=i$. The loop should be structured as *for*($j=0; j<=i; j++$). Inside the loop run an if statement *if*($j!=0$) then print star and digit else print only digit.
9. Outside the loop increment count and print a newline.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;//declaring integer variables i,j for loops , r for number of
    rows
    printf("Enter the number of rows/columns :\n");//asking user for the number
    of rows;
    scanf("%d",&r);//taking number of rows and saving in variable r
    r=r/2;
    count=r;
    for(i=0;i<r;i++) //loop for number of rows
    {
        for(j=r;j>i;j--)//loop to print digit in every column of a row
        {
            if(j!=r)
            {
                printf("%d",count);//printing digit
            }
            else
            {
                printf("%d",count);//printing digit
            }
        }
        count--;
        printf("\n");//printing newline
    }
    count++; //intialising count =3
    for(i=0;i<r;i++) //loop for number of rows
    {
        for(j=0;j<=i;j++) //loop to print digit in every column of a row
        {
            if(j!=0)
            {
                printf("%d",count);//printing digit
            }
            else
            {
                printf("%d",count);//printing digit
            }
        }
        count++; //incrementing count
        printf("\n");//printing newline
    }
}
```



```

        if(i<=(r/2)) //if condition to print the
top half
    {
        for(j=0;j<=i;j++) // loop for digits per each
row
    {
        if(j!=0)
        {
            printf("%d",i+2); //printing digits and stars
        }
        else
        {
            printf("%d",i+2); //printing digits
        }
    }
    else //else condition to print
the bottom half
    {
        for(j=i;j<r;j++) //loop for printing
    {
        if(j!=i)
        {
            printf("%d",r-i+1); //printing stars and
digit
        }
        else
        {
            printf("%d",r-i+1); //printing digit
        }
    }
    }

    printf("\n"); // printing newline
after each row
}
}

```

PRINTING PATTERN:

2

3*3

4*4*4

4*4*4

3*3

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable. ('r' in this case).
2. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=1$; $i\leq r$: $i++$).
3. Use an if condition to print the top half of the pyramid. *if* ($i\leq r/2$) run a loop from $j=0$ to $j\leq i$. The loop should be structured as *for*($j=1$; $j\leq i$; $j++$)
4. Run an if statement *if*($j!=1$). If true the print star and count else only print count.
5. Then in the outer if statement increment count. Then print a newline
6. Inside this loop print count.
7. Outside this loop increment count and print a newline
8. Else decrement count and run a loop from $j=0$ to $j<r-i$. The loop should be structured as *for*($j=0$; $j<r-i$; $j++$). inside the loop run an if statement *if*($j!=0$) then print star and count else just print count.
9. After the loop print a newline.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;                                //declaring integer
    variables i,j for loops , r for number of rows
    printf("Enter the number of rows/columns :\n");    //asking user for the
    number of rows;
    scanf("%d",&r);                                    //taking number of
    rows and saving in variable r
    count=2;                                           //initialising count =2
    for(i=1;i<=r;i++)                                //loop for number of
    rows
    {
        if(i<=r/2)
        {
            for(j=1;j<=i;j++)                        //loop to print digit
            in every column of a row
            {
                if(j!=1)
                {
                    printf("%d",count);                //printing digit
                }
            }
            else
            {
                printf("%d",count);                    //printing digit
            }
        }
    }
}
```

```

        count++;
        printf("\n");
    }
    else
    {
        count--;

        for(j=0;j<r-i+1;j++)
        {
            if(j!=0)
            {
                printf("%d",count);
            }
            else
            {
                printf("%d",count);
            }
        }

        printf("\n");
    }
}

```

//incrementing count
//printing newline

PRINTING PATTERN:

6*6*6*6

5*5*5

4*4

3

3

4*4

5*5*5

6*6*6*6

PREREQUISITE:

Basic knowledge of C language and use of loops.

ALGORITHM:

1. Take the number of rows as input from the user and store it in any variable.('r' in this case).
2. Divide the value of 'r' by 2 and replace it in r. And give this value to count and increase count by 2.
3. Run a loop 'r' number of times to iterate through each of the rows. From $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$: $i++$).
4. Run a loop from $j=r$ to $j>i$. The loop should be structured as *for*($j=r$; $j>i$; $j--$)
5. Run an if statement *if*($j!=r$). If true the print star and count else only print count.
6. Then in the outer if statement decrement count. Then print a newline
7. Outside this loop increment count and run a loop from $i=0$ to $i<r$. The loop should be structured as *for*($i=0$; $i<r$: $i++$).
8. Run a nested loop from $j=0$ to $j<=i$. The loop should be structured as *for*($j=0$; $j<=i$; $j++$). Inside the loop run an if statement *if*($j!=0$) then print star and digit else print only digit.
9. Outside the loop increment count and print a newline.

CODE IN C:

```
#include<stdio.h>
int main()
{
    int i,j,r,count;//declaring integer variables i,j for loops , r for number of
    rows
    printf("Enter the number of rows :\n");//asking user for the number of rows;
    scanf("%d",&r);//taking number of rows and saving in variable r
    r=r/2;
    count=r+2;
    for(i=0;i<r;i++) //loop for number of rows
    {
        for(j=r;j>i;j--)//loop to print digit in every column of a row
        {
            if(j!=r)
            {
                printf("%d",count);//printing digit
            }
            else
            {
                printf("%d",count);//printing digit
            }
        }
        count--;
        printf("\n");//printing newline
    }
    count++; //initialising count =3
    for(i=0;i<r;i++) //loop for number of rows
    {
        for(j=0;j<=i;j++) //loop to print digit in every column of a row
        {
            if(j!=0)
            {
                printf("%d",count);//printing digit
            }
            else
```

```
        {
            printf("%d",count);//printing digit
        }
        count++; //incrementing count
        printf("\n"); //printing newline
    }
}
```

Programming Data Types Questions :A

Question 1

What collects all the source code for an application and prepares them to be ready for execution?

- A Executor
- B Loader
- C Linker
- D Compiler

Question

2

Wrong

Usain Bolt runs 400 meters race, the timekeeper wants to write a program to save his track timing what kind of a data type should he be using to store the temporary data?

- int
- B Float
- C Pointer
- Double

Question 2 Explanation:

The double type of data will give him the highest precision. e.g – 9.59716 secs

Question 3

Wrong

Raman works in a reputed software company. His manager has asked him to make students sit in an array like position, an array which can store numbers in both negative $a[-4]$ and $a[4]$ and so on in a special compiler. He can only use 8 bytes of addressing. How many people can sit in this array and what is the last negative position?

A 128, -127

264, -128

C 264, -127

512, -127

Question 3 Explanation:

For signed integer the possibilities are $2^8 = 256$. However for unsigned we can divide this by 2 i.e. 128. Thus array according to the basic storage will be from -128 to +127.

Question 4

Wrong

Paras has to write a code using a consecutive number of registers and these registers can at max have 11 bits. How many signed numbers can he store in this?

A 1024

B 512

2048

256

Question 4 Explanation:

It is very simple since it is signed thus, $2^{11} = 2048$ will be the answer.

Question 5

Wrong

Assembler works to convert assembly language program into machine language:

Before the computer can execute it

After the computer can execute it

C In between execution

D All of these

Question 6

Wrong

Prateek has got homework from his teacher to find the numbers bits in a data type that will help me write in Portuguese Language has about 60 letters in it. What are number of bits he must be using –

A 4

B 5

6

3

Question 6 Explanation:

Since 2^6 is 64. For 60 a 6 bit integer will be enough

Question 7

Wrong

Which of the following is not a data type?

A int

B float

C short

boolean

all the above

Question 7 Explanation:

All of them are data types

Question 8

Wrong

What is the storage size and the range for short?

A 4 bytes, 65,534

B 2 bytes, 65,534

2 bytes, -32,768 to 32,767

4 bytes, -32,768 to 32,767

Question 8 Explanation:

Check Data type info here – https://www.tutorialspoint.com/cprogramming/c_data_types.htm

Question 9

Correct

Which of the following is not derived type of Data Type?

A Pointer types

B Array types

Enum Type

D Structure types

E Union types

Question 9 Explanation:

All except, Enum Type are derived type

Programming Iteration, Recursion, Decision Questions :A

Question 1

Correct

Predict the output of the questions –

```
Function main() {  
    double d = 123.4  
    static float f =123.4  
    if (m equals i)  
        print "Both of them are equal"  
    else if( f > d )  
        print "Float is greater"  
    else  
        print "Double is greater"  
}
```

A Float is greater

B Double is greater

C Both of them are equal

Code will generate error

Question 1 Explanation:

equals is not a function to compare float and double

Question 2

Wrong

As a project, Parag wants to write a code which should increment its value until a condition is satisfied. Which type of structure should he be using?

A For

B While

Do while

Perforate

Question 2 Explanation:

Do while is exactly what the questions says for loop does the same thing but not in the exact scenario as the question

Question 3

Wrong

Predict the output of the following code
`int p = 1256, q, r, s=10;q=p/s;r=p-q;print r;`

A 126

1131

C 125.6

1130.6

Question 3 Explanation:

$1256/10 = 125$
 $1256 - 125 = 1131$

Question 4

Wrong

```
function foo() {  
  int a = 0;  
  switch(a) {  
    case 0 : print "2";  
    case 2 : print "2";  
    case 4 : print "2";  
  }
```

A nothing gets printed

2

C 2

222

Question 4 Explanation:

222 will be printed as there is no break anywhere

Question 5

Wrong

Predict the output of the following Code

```
function foo()  
{  
  int a= 245,b=5, d;  
d = a/b  
  switch(d)  
  {  
    case 4 :print "I behaved correctly"  
    break  
    case 49:print "I behaved with accuracy"  
    break  
    default:print "Not Sure"  
  }  
}
```

I behaved correctly

B I behaved with accuracy

C Not Sure

Error

Question 5 Explanation:

Break doesn't have – ;

Question 6

Wrong

Which of the following is used to iteratively do the same thing again and again with auto feeding values?

For Loop

B do while

C if

Recursion

Question 6 Explanation:

Only recursion feeds value to it again and again in other loops we have to write code to iterative values with i++ or i= i-2 etc.

Question 7

Wrong

Integer a = 20, b =10, c = 20, d =10
Print a*b/c-d
Print a*b/(c-d)
Will the output be same for the two ?
The output will have a difference of 20

B Will be same

Cant be said depends on compiler

D differ by 100

Question 7 Explanation:

it is very basic

Question 8

Wrong

Predict the output of the following –

```
#include <stdio.h>

int fun(int n)
{
    if (n == 4)
        return n;
    else return 2*fun(n+1);
}

int main()
{
    printf("%d ", fun(2));
    return 0;
}

2
```

B 8

16

D 32

Question 8 Explanation:

Fun(2) | 2*fun(3) | 2*fun(4) | 4After tree evaluation we get 16 So, C is the correct answer

Question 9

Correct

What will be the output of following program ?

```
#include<stdio.h>
int main( )
{
    int a=300,b,c;
    if(a>=400)
    b=300;
    c=200;
    printf("%d,%d\n",b,c);
    return 0;
}
```

A Garbage value, Garbage Value

B 300,200

C 200,300

Garbage value,200

Question 9 Explanation:

As the condition within the if statement is false so the value of b will not be initialized so it will print the garbage value and c is initialized to 200 so the output will be Garbage,200

Question 10

Wrong

What will be the output of code?

```
#include<stdio.h>
int main( )
{
    int x=3,y,z;
    y=x=10;
    z=x<10;
    printf("x=%dy=%dz=%d\n",x,y,z);
    return 0;
}
```

10,10,10

10,10,0

C 0,0,0

D 0,10,10

Programming Procedures, functions and Scope :A

Question 1

Correct

The default parameter passing mechanism is

call by value

B call by reference

C call by value result

D None of these.

Question 2

Wrong

Determine output:

```
main()
{
    int i = 5;
    printf("%d%d%d%d", i++, i--, ++i, --i, i);
}
```

A 5454

45445

C 54554

45545

Question 3

Wrong

What will be printed when this program is executed?

```
int f(int x)
```



```

{
    if(x <= 4)
        return x;
    return f(--x);
}
void main()
{
    printf("%d ", f(7));
}

```

A 4 5 6 7

1 2 3 4

4

D Syntax error

E Runtime error

Question 3 Explanation:

In this recursive function call the function will return to main caller when the value of x is 4. Hence the output.

Question 4

Wrong

Which of the following function calculates the square of 'x' in C?

A `sqr(x)`

`power(x, 2)`

`power(2, x)`

D `pow(x, 2)`

E `pow(2, x)`

Question

5

Wrong

`char* myfunc(char *ptr) { ptr+=3; return(ptr); }void main() { char *x, *y; x = "PrepInsta"; y = myfunc(x); printf("y=%s", y); }` What will be printed when the sample code above is executed?

A y=PrepInsta

B y=pInsta

y=Insta

y=nsta

E y=epInsta

Question 6

Wrong

```
#include <stdio.h>
extern int var;
int main()
{
    var = 10;
    printf("%d ", var);
    return 0;
}
```

A 20

0

compiler error

Question 6 Explanation:

extern only defines it but allocates no memory so compiler error.

Programming Arrays, Linked Lists, Trees, Graphs Questions :A

Question 1

Wrong

In a Doubly Linked list how many nodes have atleast 1 node before and after it?

N+1

B N

N-2

D N-1

Question 1 Explanation:

The first and the last node in the doubly linked list will point to Null

Question 2

Correct

Raman is 7 years old and he wants to color a book. The book happens to be about DSA and contains a Complete binary tree with 7 levels, he wants to use different color for every tree nodes. How many colors will he need?

A 28

B 31

C 63

127

Question 2 Explanation:

to find the total no of nodes in nth level by 2^{n-1} 1 level 1 nodes, 2 level 3 nodes, 3 level 7 nodes, 4 level 15 nodes, 5 level 31 thus 7 has 127 nodes

Question 3

Wrong

He again gets a book with x number of non-leaf nodes. How many total number of nodes will be there for him color

A $2x$

B $x + 1$

$\log x$

$2x + 1$

Question 4

Wrong

Which of the following is true about bipartite Graph?

no cycle of odd length

B $n^{\log n}$

C n edges

a cycle of odd length

Question 5

Wrong

What is Dynamic Allocation in Array?

A Allocation that takes place at compile time

B Allocation that take place as bipartite graph

memory allocation that takes place during run time rendering the resizing of an Array

All of these

Question 6

Wrong

Raman is 7 years old and he wants to colour a book. The book happens to be about DSA and contains 8 vertices in an undirected graph with 7 levels, he wants to use a different colour for every node. How many colors will he need?

A 63

B 63

28

32

Question 6 Explanation:

The formulae is $n*(n-1)/2$

Question 7

Wrong

The height of a BST is given as h . Consider the height of the tree as the no. of edges in the longest path from root to the leaf. The maximum no. of nodes possible in the tree is?

A $2^{h-1} - 1$

$2^{h+1} - 1$

$2^h + 1$

D $2^{h-1} + 1$

Question 8

Wrong

Which type of traversal of binary search tree outputs the value in sorted order?

A Post-order

B Pre-order

In-order

None

Question 8 Explanation:

Inorder gives in correct order

Question 9

Wrong

the run time for traversing all the nodes of a binary search tree with n nodes and printing them in an order is a) b) c) d)

$O(n)$

B $O(\sqrt{n})$

C $O(\log(n))$

$O(n \lg(n))$

Question 10

Wrong

A binary search tree is generated by inserting in order the following integers: 50, 15, 62, 5, 20, 58, 91, 3, 8, 37, 60, 24. The number of the node in the left sub-tree and right sub-tree of the root, respectively, is

A (3, 8)

B (8, 3)

(7, 4)

(4, 7)

Cts

You have to classify a string as “GOOD”, “BAD” or “MIXED”. A string is composed of lowercase alphabets and ‘?’. A ‘?’ is to be replaced by any of the lowercase alphabets. Now you have to classify the string on the basis of some rules. If there are more than 3 consonants together, the string is considered to be “BAD”. If there are more than 5 vowels together, the also string is considered to be “BAD”. A string is “GOOD” if its not “BAD”. Now when question marks are involved, they can be replaced with consonants or vowels to make new strings. If all the choices lead to “GOOD” strings then the input is considered as “GOOD”, and if all the choices lead to “BAD” strings then the input is “BAD”, else the string is “MIXED”?

LOGIC

1. Convert the string in **0's and 1's**. All consonants will be considered as 1 and vowels as 0.
2. Make 2 D array (this is dynamic programming approach) and start matching.
3. For string matching, one side will be the string (converted with 0's and 1's) and another side with vowels and consonants as 0 and 1.

cons & Vow/ String		W	A	Y	T	O	C	R	A	C	K
		0	1	0	1	1	0	1	1	0	1
0		0	0	1	0	0	1	0	0	1	0
1		0	1	0	1	2	0	1	2	0	1

1. Whenever the 0 will be matched with 0, increment previous array element by 1. Similarly when 1 will be matching with 1, increment previous array element by 1.
2. In case of 3 consecutive consonants, the current array value reaches to 3 or 5 consecutive vowels the string is said to “**BAD**”.

Now here comes ? marks case

cons & Vow/ String		W	A	Y	T	?	C	R	A	C	K
		0	1	0	1	1	?	1	1	0	1
0		0	0	1	0	0	1	0	0	1	0
1		0	1	0	1	2	3	4	5	0	1

In case of question mark “?” add one to both consonant and vowel array. Also, make a Boolean flag which will be true in case of “?” occurs. If “?” and 3 consecutive consonants or 5

consecutive vowels occurred then the string is said to be **“MIXED”**, otherwise the string is **“GOOD”**.

Please see below code for clear understanding, some of the corner test cases are tested which are also commented there, to test uncomment it and run.

Some more test cases are also given at the end of this post.

Code –

Find if string is good, bad or mixed Code

```
#include "string.h"
#include "iostream"
using namespace std;

//alphabet check
bool alphabetcheck(char alphabet){
    if (alphabet >= 'a' && alphabet <= 'z')
        return true;
    return false;
}

//vowel check
int vowelcheck(char vowel)
{
    if (vowel == 'a' || vowel == 'e' || vowel == 'i' || vowel == 'o' ||
vowel == 'u')
        return 0;
    return 1;
}

int main(void) {
    //the two sequences
    //string X = "waytocrack";// ---- GOOD
    //string X = "waaaaaaytocrack";//--BAD
    string X = "wayt?arack"; // mixed
    //length of the sequences
    int XLen = X.size();
    int Arr[2][20];
    memset(Arr, 0, sizeof(Arr[0][0]) * 2 * 20);
    int max0 = 0;
    int max1 = 0;
    int index;
    bool mixed_value = false;
    for (size_t i = 0; i < XLen; i++)
    {
        //alphabet check
        if (alphabetcheck(X[i]))
        {
```



```

//vowel check fucntion:
if ((vowelcheck(X[i])) == 0)
{
    Arr[0][i+1] = Arr[0][i] + 1;
    if (Arr[0][i + 1] > max0)
        max0 = Arr[0][i + 1]; //check for max 0
    if ((mixed_value == true) && (max0 >= 5) && (Arr[0][i + 1] >=
5))
// Arr[0][i + 1] >= 5 when current value is greater than 5
    {
        if ((i - index >= 5 || (Arr[1][index] + Arr[0][index + 5]
== 7)))
            mixed_value = false;
    }
}
else
{
    Arr[1][i + 1] = Arr[1][i] + 1;
    if (Arr[1][i + 1] > max1)
        max1 = Arr[1][i + 1]; //check for max 1
    if (mixed_value == true && max1 >= 3 && Arr[1][i + 1] >= 3 )
// Arr[0][i + 1] >= 5 when current value is greater than 3
    {
        if ((i - index >= 3 || (Arr[0][index] + Arr[1][index + 3]
== 7)) )
            mixed_value = false;
    }
}
if (mixed_value == false && (max0 >= 5 || max1 >= 3)){
//checking the count value GREATER than or 3
    cout << " BAD " << endl;
    exit(0);
}

}

else if (X[i] == '?'){
    //increament both o count and 1 count.
    Arr[0][i + 1] = Arr[0][i] + 1;
    if (Arr[0][i + 1] > max0)
        max0 = Arr[0][i + 1]; //check for max 1
    Arr[1][i + 1] = Arr[1][i] + 1;
    if (Arr[1][i + 1] > max1)
        max1 = Arr[1][i + 1]; //check for max 1
    index = i;
    mixed_value = true;
}

}

if (mixed_value & (max0 >= 5 || max1 >= 3))
    cout << " MIXED " << endl;
else cout << " GOOD " << endl;

return 0;
}

```

These are some corner test cases:

a?fafff	BAD
??aa??	MIXED
abc	GOOD
aaa?aaafff	BAD
aaaa?ff?aaa?aaa?fff	BAD
aaaaff?	MIXED
aaaaf?	GOOD
?aaaaffaaf?aaaafff	BAD
?aaaaffaaf?aaaaff	MIXED
vaxaaaa?bbadadada	BAD
aaaa?bb	BAD
vabb?aaaadadada	BAD
vabab?aaaadadada	MIXED

The solution is 3 or more than 3 consecutive consonants and 5 and more than 5 consecutive vowels.

You can check some of the tested test cases at the end of the code.

We are given an array with n elements from {1,2,3,4}. Find the number of minimum changes required to be performed so that no two adjacent numbers are same?

Please write the code in the comments? We will add them here.

We are given a count of songs to be played – n, highest volume allowed – h, initial volume – i, and list of allowed volume change A[] of size n. The singer can either increase/decrease the volume of sound system for the next song by the allowed volume change A[j] for jth song from the volume of the j-1th song. The aim is to maximize the volume of last sound. Find the maximum volume that can be attained, or return -1 if there is no possibility of changing volume due to the given constrains. (Volume cannot be in negative.)

Please write the code in the comments? We will add them here.

Write a program to print all Subsequences of String which Start with Vowel and End with Consonant

Given a string return all possible subsequences which start with vowel and end with a consonant. A String is a subsequence of a given String, that is generated by deleting some character of a given string without changing its order.

Examples:

Input : 'abc'

Output : ab, ac, abc

Input : 'aab'

Output : ab, aab

Explanation of the Algorithm:

Step 1: Iterate over the entire String

Step 2: check if the i th character for vowel

Step 3: If true iterate the string from the end,
if false move to next iteration

```
Step 4: check the jth character for consonant
        if false move to next iteration
        if true perform the following
```

Step 5: Add the substring starting at index i and ending at index j to the hastset.

Step 6: Iterate over the substring drop each character and recur to generate all its subString

```
// Java Program to generate all the subsequence
```

```
// starting with vowel and ending with consonant.
```

```
import java.util.HashSet;
```

```
public class Subsequence {
```

```
// Set to store all the subsequences
```

```
static HashSet<String> st = new HashSet<>();
```

```
// It computes all the possible substring that
```

```
// starts with vowel and end with consonant
```

```
static void subsequence(String str)
```

{

```
// iterate over the entire string
```

```
for (int i = 0; i < str.length(); i++) {
```

```
// test ith character for vowel
```

```
if (isVowel(str.charAt(i))) {
```

```
// if the ith character is vowel
```

```
// iterate from end of the string
```

```
// and check for consonant.
```

```
for (int j = (str.length() - 1); j >= i; j--) {
```

```
// test jth character for consonant.
```

```
if (isConsonant(str.charAt((j)))) {
```

```
// once we get a consonant add it to
```

```

        // the hashset
        String str_sub = str.substring(i, j + 1);
        st.add(str_sub);
        // drop each character of the substring
        // and recur to generate all subsequence
        // of the substring
        for (int k = 1; k < str_sub.length() - 1; k++) {
            StringBuffer sb = new StringBuffer(str_sub);
            sb.deleteCharAt(k);
            subsequence(sb.toString());
        }
    }
}

// Utility method to check vowel
static boolean isVowel(char c)
{
    return (c == 'a' || c == 'e' || c == 'i' || c == 'o'
            || c == 'u');
}

// Utility method to check consonant
static boolean isConsonant(char c)
{
    return !(c == 'a' || c == 'e' || c == 'i' || c == 'o'
            || c == 'u');
}

// Driver code
public static void main(String[] args)
{
    String s = "xabcef";
    subsequence(s);
    System.out.println(st);
}
}

```

Output:

```
[ef, ab, ac, aef, abc, abf, af, acf, abcef, abcf, acef, abef]
```

Ques. Find winner of an election where votes are represented as candidate names. Given an array of names of candidates in an election. A candidate name in array represents a vote casted to the candidate. Print the name of candidates received Max vote. If there is tie, print lexicographically smaller name. A **simple solution** is to run two loops and count occurrences of every word. Time complexity of this solution is $O(n * n * \text{MAX_WORD_LEN})$. An **efficient solution** is to use [Hashing](#). We insert all votes in a hash map and keep track of counts of different names. Finally we traverse the map and print the person with maximum votes.

```

// Java program to find winner in an election.
import java.util.*;

```

```

public class ElectoralVotingBallot
{
    /* We have four Candidates with name as 'John',
    'Johnny', 'jamie', 'jackie'.
    The votes in String array are as per the
    votes casted. Print the name of candidates
    received Max vote. */
    public static void findWinner(String votes[])
    {
        // Insert all votes in a hashmap
        Map<String,Integer> map =
            new HashMap<String, Integer>();
        for (String str : votes)
        {
            if (map.keySet().contains(str))
                map.put(str, map.get(str) + 1);
            else
                map.put(str, 1);
        }
        // Traverse through map to find the candidate
        // with maximum votes.
        int maxValueInMap = 0;
        String winner = "";
        Map.Entry<String,Integer> entry;
        for (entry : map.entrySet())
        {
            String key = entry.getKey();
            Integer val = entry.getValue();
            if (val > maxValueInMap)
            {
                maxValueInMap = val;
                winner = key;
            }
            // If there is a tie, pick lexicographically
            // smaller.
            else if (val == maxValueInMap &&
                winner.compareTo(key) > 0)
                winner = key;
        }
        System.out.println("Winning Candidate is : " +
            winner);
    }
    // Driver code
    public static void main(String[] args)
    {
        String[] votes = { "john", "johnny", "jackie",
            "johnny", "john", "jackie",
            "jamie", "jamie", "john",
            "johnny", "jamie", "johnny",
            "john" };
        findWinner(votes);
    }
}

```

Output:
John

A man starts from his house with a few pan cakes. let they be N. Now he visits K places before reaching home. At each place he can buy a cake, sell a cake or do nothing. But he must sell L cakes before reaching home. Find the maximum number of cakes he can have at any point in his journey. N, K, L are given as input?

Please comment down the solutions, we will add them here.

Find the row with maximum number of 1s?

Given a boolean 2D array, where each row is sorted. Find the row with the maximum number of 1s.

```
Example
Input matrix
0 1 1 1
0 0 1 1
1 1 1 1 // this row has maximum 1s
0 0 0 0
```

Output: 2

A simple method is to do a row wise traversal of the matrix, count the number of 1s in each row and compare the count with max. Finally, return the index of row with maximum 1s. The time complexity of this method is $O(m*n)$ where m is number of rows and n is number of columns in matrix.

We can do better. Since each row is sorted, we can **use Binary Search** to count of 1s in each row. We find the index of first instance of 1 in each row. The count of 1s will be equal to total number of columns minus the index of first 1.

See the following code for implementation of the above approach.

```
#include <stdio.h>
#define R 4
#define C 4
/* A function to find the index of first index of 1 in a boolean array arr[]
*/
int first(bool arr[], int low, int high)
{
    if(high >= low)
    {
        // get the middle index
        int mid = low + (high - low)/2;
        // check if the element at middle index is first 1
        if ( ( mid == 0 || arr[mid-1] == 0) && arr[mid] == 1)
            return mid;
        // if the element is 0, recur for right side
        else if (arr[mid] == 0)
            return first(arr, mid + 1, high);
        // if the element is 1, recur for left side
        else if (arr[mid] == 1)
            return first(arr, low, mid - 1);
    }
    return -1;
}
```

```

        return first(arr, (mid + 1), high);
    else // If element is not first 1, recur for left side
        return first(arr, low, (mid - 1));
    }
    return -1;
}
// The main function that returns index of row with maximum number of 1s.
int rowWithMax1s(bool mat[R][C])
{
    int max_row_index = 0, max = -1; // Initialize max values
    // Traverse for each row and count number of 1s by finding the index
    // of first 1
    int i, index;
    for (i = 0; i < R; i++)
    {
        index = first (mat[i], 0, C-1);
        if (index != -1 && C-index > max)
        {
            max = C - index;
            max_row_index = i;
        }
    }
    return max_row_index;
}
/* Driver program to test above functions */
int main()
{
    bool mat[R][C] = { {0, 0, 0, 1},
                        {0, 1, 1, 1},
                        {1, 1, 1, 1},
                        {0, 0, 0, 0}
    };
    printf("Index of row with maximum 1s is %d n", rowWithMax1s(mat));
    return 0;
}

```

Output:

Index of row with maximum 1s is 2

Time Complexity: $O(m \log n)$ where m is a number of rows and n is a number of columns in a matrix.

1. [Program to check if a Binary tree is BST or not](#)
2. [Find the Lowest Common Ancestor in a Binary Search Tree](#)
3. [Program to check if a binary tree is height balanced or not](#)
4. [Program Two Nodes of a BST are Swapped correct the BST](#)
5. [Find maximum in binary tree](#)
6. [Program to Connect nodes of a Tree at same level](#)
7. [Find the Lowest Common Ancestor in a Binary Tree](#)
8. [Add Two Numbers Represented by Linked Lists](#)
9. [Detecting Loop in a Linked List](#)

