**PATTERN PROBLEMS**

**Pattern 1:**



**CODE:**

void printSquare(int n) {

for( int j = 0 ; j < n ; j++ )

{

for( int i = 0 ; I < n ; i++ )

{

cout << "\* ";

}

cout << endl;

}

}

}

**Pattern 2:**



**CODE:**

void printTriangle(int n) {

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

{

for ( int j = n-i+1 ; j <= n ; j++ ) // Or use j = 1 to i

{

cout << "\* ";

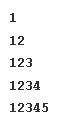
}

cout << endl;

}

}

**Pattern 3:**



**CODE:**

void printTriangle(int n) {

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

{

for ( int j = 1 ; j <= i; j++ )

{

cout << j << ' ';

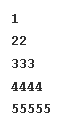
}

cout << endl;

}

}

**Pattern 4:**



**CODE:**

void printTriangle(int n) {

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

{

for( int j = 1; j <= i ; j++ )

{

cout << i << ' ' ;

}

cout << endl;

}

}

**Pattern 5:**



**CODE:**

void printTriangle(int n) {

//using while loops, not necessary

while( n != 0 )

{

int i = 1;

while( i <= n )

{

cout << "\* ";

i++;

}

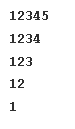
cout << endl;

n--;

}

}

**Pattern 6:**



**CODE:**

void printTriangle(int n){

for (int i = n ; i >= 1 ; i--)

{

for ( int j = 1 ; j <= i ; j++ )

{

cout << j << ' ';

}

cout << endl;

}

}

**Pattern 7:**



**CODE:**

void printTriangle(int n) {

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

{

for ( int j = n-i ; j >=1 ; j-- )

{

cout << ' ';

}

for ( int j = 1 ; j <= 2\*i - 1 ; j++ )

{

cout << '\*';

}

cout << endl;

}

}

**Pattern 8:**



**CODE:**

void printTriangle(int n) {

// code here

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

{

for ( int j = 1 ; j <= i - 1 ; j++ )

{

cout << ' ';

}

for ( int j = 1 ; j <= 2\*n - 2\*i + 1 ; j++)

{

cout << "\*" ;

}

cout << endl;

}

}

**Pattern 9:**



**CODE:**

void printDiamond(int n) {

// alternatively split the problem by top and bottom part and combine solutions

// sometimes combining solutions is easier and more efficient

for ( int i = 0 ; i < 2\*n ; i++ )

{

int x = n - i - 1,limit = i+1;

if ( i >= n)

{

x = i - n;

limit = 2\*n-i; //5

}

//space

for ( int j = 1 ; j <= x ; j++) // j <=1

{

cout << ' ';

}

//pattern-stars

for (int j = 0 ; j < limit ; j++ )

{

cout << "\* ";

}

cout << endl;

}

}

**Pattern 10:**



**CODE:**

void printTriangle(int n) {

// code here

for ( int i = 0 ; i < 2\*n - 1 ; i++ )

{

int limit = i+1;

if( i >= n)

{

limit = 2\*n -i - 1;

}

for ( int j = 0 ; j < limit ; j++ )

{

cout << "\* ";

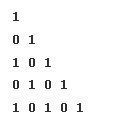
}

cout << endl;

}

}

**Pattern 11:**



**CODE:**

void printTriangle(int n) {

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

{

for (int j = 0 ; j <= i ; j++ )

{

if ( (i+j) %2 == 0 )

{

cout << "1 ";

}

else

{

cout << "0 ";

}

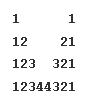
}

cout << endl;

}

}

**Pattern 12:**



**CODE:**

void printTriangle(int n) {

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

{

// first triangle number pattern

for ( int j = 0 ; j <= i ; j++ )

{

cout << j+1 << ' ';

}

// first space

for ( int j = 0 ; j < 2\*(n - 1 - i); j++ )

{

cout << " ";

}

// second number

for ( int j = i ; j >= 0 ; j-- )

{

cout << j+1 << ' ';

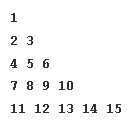
}

cout << endl;

}

}

**Pattern 13:**



**CODE:**

void printTriangle(int n) {

int cnt = 1;

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

{

for ( int j = 1 ; j <= i ; j++)

{

cout << cnt++ << ' ';

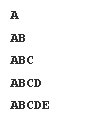
}

cout << endl;

}

}

**Pattern 14:**



**CODE:**

void printTriangle(int n) {

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

{

char ch = 65;

for ( int j = 1 ; j <= i ; j++ )

{

cout << ch++;

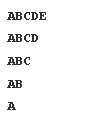
}

cout << endl;

}

}

**Pattern 15:**



**CODE:**

void printTriangle(int n) {

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

{

char ch = 65;

for ( int j = n-i+1 ; j >= 1 ; j-- )

{

cout << ch++;

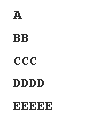
}

cout << endl;

}

}

**Pattern 16:**



**CODE:**

void printTriangle(int n) {

char ch = 65;

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

{

for ( int j = 1 ; j <= i ; j++ )

{

cout << ch;

}

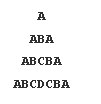
ch++;

cout << endl;

}

}

**Pattern 17:**



**CODE:**

void printTriangle(int n) {

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

{

char ch = 65;

// space

for ( int j = 1 ; j <= n-i ; j++ )

{

cout << ' ';

}

// alphabets

for ( int j = 1 ; j <= 2\*i - 1 ; j++ )

{

cout << ch;

if ( j >= i )

{

ch = ch - 1;

}

else

{

ch = ch + 1;

}

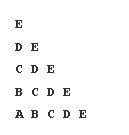
}

cout << endl;

}

}

**Pattern 18:**



**CODE:**

void printTriangle(int n) {

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

{

char ch = 64 + n;

for ( int j = 1 ; j <= i ; j++ )

{

cout << ch-- << ' ';

}

cout << endl;

}

}

**Pattern 19:**



**CODE:**

void printTriangle(int n) {

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

{

// split problem into 2x3 parts: 1.first half stars 2. space 3. second half stars

// first part

for ( int j = 1 ; j <= n - i + 1; j++ )

{

cout << '\*';

}

// space

for ( int j = 1 ; j <= 2\*(i - 1) ; j++ )

{

cout << ' ';

}

// second part

for ( int j = 1 ; j <= n - i + 1 ; j++ )

{

cout << '\*';

}

cout << endl;

}

// symmterical half done

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

{

for ( int j = 1 ; j <= i ; j++ )

{

cout << '\*';

}

for ( int j = 1 ; j <= 2\*(n - i) ; j++ )

{

cout << ' ';

}

for ( int j = 1 ; j <= i ; j++ )

{

cout << '\*';

}

cout << endl;

}

}

**Pattern 20:**



**CODE:**

void printTriangle(int n) {

for ( int i = 1 ; i <= 2\*n - 1 ; i++ )

{

int limit = i ;

if ( i > n )

{

limit = 2\*n - i;

}

// stars

for ( int j = 1 ; j <= limit ; j++ )\

{

cout << '\*';

}

// space

for ( int j = 1 ; j <= 2\*( n - limit ) ; j++ )

{

cout << ' ';

}

// stars

for ( int j = 1 ; j <= limit ; j++ )

{

cout << '\*';

}

cout << endl;

}

}

**Pattern 21:**



**CODE:**

void printSquare(int n) {

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

{

for ( int j = 1 ; j <=n ; j++ )

{

if ( i == 1 || i == n || j == 1 || j == n )

{

cout << '\*';

}

else

{

cout << ' ';

}

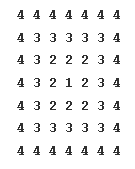
}

cout << endl;

}

}

**Pattern 22:**



**CODE:**

void printSquare(int n) {

for ( int i = 1 ; i <= 2\*n - 1 ; i++ )

{

for ( int j = 1 ; j <= 2\*n -1 ; j++ )

{

// find distance of a point to all of its edges

// take minimum edges and form matrix

// subtract n from each point and print absolute value

int top,left,right,bottom;

top = i - 1;

left = j - 1;

right = 2\*n - 1 - j;

bottom = 2\*n - 1 - i;

cout << abs( n - min( min( top, bottom ), min( right, left ) ) ) << ' ';

}

cout << endl;

}

}