CODE FOUNDATION

Pattern Printing

We will practice some of the patterns so that we become familiar with loops.

Try coding the first 3 programs yourself.

Program 1:

links: ideone link, sapphire link

Write a program to find the sum of all even numbers upto n.

Approach:

We have to check for all the numbers till n, if they are even or not. If a number is even then we have to include it in our sum. So the first thing that comes to our mind is loops. With the help of loops we can iterate till n and for each number we can check if it's even or not. Also we know that the alternating numbers from 2 are even. Suppose n=10. Now the sum of even numbers till 10 is **2+4+6+8+10=30**

```
1. #include <iostream>
2. using namespace std;
3.
4. int main() {
5.       // your code goes here
6. int n;
7. cin>>n;
8. int i=2,sum=0;
9. while(i<=n)
10.   {
11.
12.   sum+=i;

13.   i+=2;
14.   }
15.   cout<<sum<<endl;
16.      return 0;
17.   }
```

Program 2: **links:** ideone link, sapphire engine link

Write a program to print a table of n.

Approach:

Think of how you can print the table of a number. Suppose the number is 3, then you have to print like:

3 x 1= 3

3 x 2 = 6

3 x 3 = 9

If you have noticed we have to multiply 3 with all the numbers starting from 1 to 10. Using a loop we can iterate over all the numbers from 1 to 10 and multiply n with it.

```
1. #include <iostream>
2. using namespace std;
3.
4. int main() {
5.       // your code goes here
6. int n;
7. cin>>n;
8. int i;
9. for(i=1;i<=10;i++)
10.   {
11.   cout<<n<<" "<<'x'<<" "<<i<<" "<<'='<<" "<<n*i<<endl;
12.   }
13.      return 0;
14.   }
```

Output for n=3

```
3 x 1 = 3

3 x 2 = 6

3 x 3 = 9
```

```
3 x 4 = 12

3 x 5 = 15

3 x 6 = 18

3 x 7 = 21

3 x 8 = 24

3 x 9 = 27

3 x 10 = 30
```

Program 3:

link: ideone link, sapphire engine link

Find the nth fibonacci number. Note that the first fibonacci number is 0 and the 2nd fibonacci number is 1. 0, 1, 1, 2, 3, 5, 8, 13,…… is the fibonacci series.

Approach:

Notice that the first two terms of the fibonacci series are 0,1. Then the third term is 1 i.e 0+1. The fourth term is 2 i.e 1+1. The fifth term is 3, i.e 1+2. We can conclude that the nth term is the sum of previous two terms.

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.          // your code goes here
6.  int n;
7.  cin>>n;
8.  int i,first=0,second=1,next=0;
9.  for(i=3;i<=n;i++)
10. {
11. next=first+second;
12. first=second;
13. second=next;
14. }
```

```
15. cout<<next<<endl;
16.             return 0;
17. }
```

Output for n=5

6

Program 4:

Link : sapphire engine link ,ideone link

Print the given pattern.

For n=6

```
1.          *
2.         **
3.        ***
4.       ****
5.      *****
6.     ******
```

How do we approach the above program? Let's first observe the pattern.
Notice we have to print n rows. For an example here we have to print for n=6.
Before printing the stars of a row we have to print the spaces. For row_1 there are 5 spaces(i.e 6-1), for row_2 there are 4 spaces (i.e 6-2) and so on.
For row_1 we have to print only one star('*'). For row_2 we have to print 2 stars. For row_3 we have to print 3 stars. Similarly for row_n we have to print 6 stars.
Now after observing the pattern we have come to a conclusion that for a row we first have to print the spaces and after the spaces we have to print the stars. In a row there will be **(n-i) spaces** and **'i' no of stars**. Using these observations we can print the pattern.

```
7.  #include <iostream>
8.  using namespace std;
9.
10. int main() {
```

```
11.  int n;
12.  cout << "Enter n: " << endl;
13.  cin >> n;
14.  for (int i=1; i<=n; i++) {
15.      for (int j=n-i; j>=1; j--)
16.          cout << " ";
17.      for (int j=1; j<=i; j++)
18.          cout << "*";
19.      cout << endl;
20.  }
21.  return 0;
22. }
```

Standard Output

```
Enter n: 6
     *
    **
   ***
  ****
 *****
******
```

Program 5: LINK : sapphire engine link, ideone link

Print the given pattern

For n=6

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

Approach :

If we observe the pattern closely we can see row 1 has 5 spaces i.e(6-1). Row 2nd has 4 spaces, i.e (6-2). Row 3rd has 3 spaces, i.e (6-3). We can generalise this formula. In the nth row there will be (total rows - n) spaces. Similarly, we can observe a pattern for stars also. In row 1, only one star is printed i.e (2*1-1) stars. In row 2, three stars are printed i.e (2*2-1) stars. In row 3, five stars are printed i.e(2*3-1)stars. Generalising the formula, we can say that in nth row we have to print (2*n-1) stars.

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4. 
5.     int n,j,row,spaces,s;
6.     cin>>n;
7.     row=1;
8.     while(row<=n)
9.     {
10.          spaces=1;
11.          j=1;
12.          while(spaces<=n-row)
13.          {
14.           cout<<" ";
15.           spaces++;
16.          }
17.      while(j<=2*row-1)
```

```
18.         {
19.             cout<<"*";
20.             j++;
21.         }
22.         cout<<"\n";
23.         row++;
24.       }
25.   }
26.
```

Program 6 : link : sapphire engine link , ideone link

Print the following pattern:

For n=4

1

21

321

4321

Approach:

Observing the given pattern we get that the first row starts from 1. The second row starts from 2. Third row starts from 3 and so on. So the nth row starts from n. Now for each row we have to print all the numbers till 1. So we will use two loops, one for accessing the rows and other for printing the numbers in each row. Notice that in each row the numbers are decrementing till 1.

```
1. #include<iostream>
2. using namespace std;
3. int main()
```

```
4. {
5.     int n,j;
6.     int i=1;
7.     cin>>n;
8.     while(i<=n){
9.      j=i;
10.       while(j>=1){
11.
12.           cout<<j;
13.           j--;
14.       }
15.           cout<<endl;
16.           i++;
17.       }
18.       }
```

Program 7:

ProgramLink(Sapphire Engine): https://sapphireengine.com/@/eesno2

ProgramLink(IDEONE): https://ideone.com/ckaus/patterns

```
/* Print the pattern:
* * * * *   * * * * *   // 2 spaces
* * * *       * * * *   // 6 spaces
* * *           * * *   // 10 spaces
* *               * *   // 14 spaces
*                   *   // 18 spaces
*                   *   // 18 spaces
* *               * *   // 14 spaces
* * *           * * *  // 10 spaces
* * * *       * * * *  // 6 spaces
* * * * *   * * * * *  // 2 spaces
```

```
*/
#include <iostream>
using namespace std;

int main() {
   int n;
   cout << "Enter n: " << endl;
   cin >> n;

   for (int i=1; i<=n; i++) {
      for (int j=1; j<=n-i+1; j++)
         cout << "* ";
      for (int j=1; j<=2*(2*i-1); j++)
         cout << " ";
      for (int j=n-i+1; j>0; j--)
         cout << "* ";
      cout << endl;
   }

   for (int i=1; i<=n; i++) {
      for (int j=i; j>=1; j--)
         cout << "* ";
      for (int j=1; j<=2*(2*(n-i+1)-1); j++)
         cout << " ";
      for (int j=i; j>=1; j--)
         cout << "* ";
      cout << endl;
   }

   return 0;
}
```

Approach: After observing the pattern carefully, you will notice that for n = 5, firstly, we are printing 10 rows in total.After that, we divide the problem into two sub-patterns, i.e., one in which we are printing 10(5 + 5), 8(4 + 4), 6(3 + 3), 4(2 +

2) and 2(1 + 1) stars(*) and the other one is just its mirror image.Now, we will discuss the first sub-pattern as the other one can be solved similarly.

Note that firstly we are printing some stars and then some spaces and then again stars.Notice the number of spaces that we have to put in between.These spaces are varying like this: 2, 6, 10, 14, 18. We will now try to think if we can formulate these spaces in terms of the row number.

Let us denote the rows by i.

Row number(i)	**Stars(*) before**	**Spaces**	**Stars(*) after**	**Formulation for spaces**
1	5	2	5	2*(2*(1) - 1) = 1
2	4	6	4	2*(2*(2) - 1) = 6
3	3	10	3	2*(2*(3) - 1) = 6
4	2	14	2	2*(2*(4) - 1) = 6
5	1	18	1	2*(2*(5) - 1) = 6

We can clearly see from the table that for row number i, we have to print 2*(2*i-1) spaces in between. Also, you can see that for row i, we have to print n - i + 1 stars before and after the spaces.