CODE FOUNDATION

Switch Statement:

Switch case statements can be used instead of long if statements.
In switch statement we test a variable for equality against list of values.

```
switch(expression)
{
case value1:
        Statement to be executed if expression evaluates to value1;
        break;
case value2:
         Statement to be executed if expression evaluates to value2;
        break;
case value3:
         Statement to be executed if expression evaluates to value3;
        break;
default:
        Statement to be executed if expression doesn't match any case;
        break;
}
```

Note :

- **The expression provided in the switch should result in a constant value otherwise it would not be valid.**
  E.g switch(3+2) is valid but switch(x+y) is invalid
  Variable expressions are allowed provided they are assigned with fixed values.
  For example
  int x=2;
  switch(x) is valid.
- **There must be a constant-expression for a case.**
  Example

```
cons int i;
int x=5;
switch(x)
{
case i: statement1;
            break;
}
```

- **There should not be duplicate cases.**
- **We use break statements to terminate a statement sequence.If we don't use them execution will continue on into the next case. This condition is called fall through. All the cases are executed until a break statement is reached.**
- **It is not necessary to use the default statement but it's a good practice to include them.**

ProgramLink(SapphireEngine):https://sapphireengine.com/@/flhqgd
ProgramLink(IDEONE):https://ideone.com/ckaus/switch-case

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

int main()
{
        char ch;
   cout << "Enter an alphabet: " << endl;
   cin >> ch;
   switch (ch) {
      case 'a':
      case 'A':
         cout << "Vowel" << endl;
         break;
      case 'e':
      case 'E':
         cout << "Vowel" << endl;
         break;
      case 'i':
      case 'I':
         cout << "Vowel" << endl;
         break;
      case 'o':
      case 'O':
         cout << "Vowel" << endl;
         break;
```

```
26.       case 'u':
27.       case 'U':
28.          cout << "Vowel" << endl;
29.          break;
30.       default:
31.          cout << "Consonant" << endl;
32.    }
33.           return 0;
34. }
```

Standard Output

```
Enter an alphabet: u
Vowel
```

LOOPS:

In our day to day life if we have to do a task repetitively what do we do? We find an easier way to do it or we just devise a way to minimize the repetitive work. Similarly if we have to do certain operations that do the same thing repetitively, we don't write the set of statements again and again. Instead we use loops. For example, if we have to print the value of a variable 1000 times, instead of writing the cout statement 1000 times, we will use loops.Inside the loop we will write the cout statement only once. Depending upon a certain condition we can execute a number of statements several times using loops.
Loop consists mainly of 3 statements: initialization, condition, increment/decrement.

While loop

In the while loop a condition or test expression is tested. If the condition is true then all the statements are executed. This continues till the condition becomes false.

Syntax:

```
while( test_ expression )
{
```

```
        //statements to be executed;
        //increment or decrement
}
//statement after the loop
```

Step1: Check the test_expression.
Step2: If it evaluates to true then all the statements inside the while body will get executed. If it evaluates to false then the control comes out of the loop and the statements just after the loop are executed.
Step3: After executing statements inside the while loop it will again check for the test_expression and the process continues.

Example:

```
int i=1; // This is initialisation
while(i<=5)  // i<=5 is the condition or test_expression
{
        cout<<"Hi"<<endl;
        i++;          // This is increment
}
cout<<"Control outside while loop"<<endl;
```

The above example prints "Hi" five times. In the starting we initialise the variable i to 1 then we check whether i<=5? If the condition is true then statements inside the while loop body are executed. And the process continues.

ProgramLink(Sapphire Engine): https://sapphireengine.com/@/6p8i1a
ProgramLink(IDEONE): https://ideone.com/ckaus/loops

Program:

```
1.  /* Printing the answer of the series: 1/1! + 2/2! + 3/3! + ... + n/n! */
2.  #include <iostream>
3.  using namespace std;
4.
5.  int main() {
6.
7.     int n;
8.     cout << "Enter the value of n: " ;
9.     cin >> n;
```

```
10.    double series = 0;
11.    double fact = 1;
12.    int r = 1;      // initialisation
13.    while(r <= n) { // checks condition
14.        series +=  r / fact;
15.        fact *= (r + 1);
16.        r++;       // updating the loop-control variable
17.    }
18.    cout << "Answer: " << series << endl;
19.
20.    return 0;
21. }
```

Standard Output

Enter the value of n: 10

Answer: 2.71828

FOR LOOP

In the for loop first initialise a variable to a value. Then check for the condition. If the test expression evaluates to true then the statements inside the for loop body are executed and after that the loop variable gets updated.

Syntax:

```
for( initialisation; test_expression; increment/decrement)
{
        //statements to be executed;
}
```

we can write the above code using for loop also.

```
for( int i=1; i<=5; i++)  //int i=1; is initialisation , i<=5 is test expression , i++ is increment
{
        cout<<"Hi"<<endl;
}
cout<<"Control outside while loop"<<endl;
```

Note:
We can also initialise the variable outside for loop
E.g

```
 int i=1;
for(;i<=5;i++)
{
cout<<i<<endl;
}
```

Both for loop and while loop are entry controlled loops, which means in both of them we check for the test expression before executing the statements.

ProgramLink(Sapphire Engine): https://sapphireengine.com/@/d01h17
ProgramLink(IDEONE): https://ideone.com/ckaus/loops

Program:

```
1. #include <iostream>
2. using namespace std;
3.
4. int main() {
5.
6.    int n;
7.    cout << "Enter n: ";
8.    cin >> n;
9.    // multiple initialisations, conditions and
  incrementations are allowed in for loops
10.     for (int i=1, j=2, k=3; i <= n && j <= n+1 && k <=
  n+2; i++, j++, k++)
11.         cout << i << " " << j << " " << k << endl;
12.
13.     return 0;
14.  }
```

Standard Output

```
Enter n: 5
1 2 3
2 3 4
3 4 5
4 5 6
5 6 7
```

The do-while Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax :

```
do {
  // code block to be executed
}while (condition);
```

The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example

```
int i = 0;

do {
  cout << i << "\n";
  i++;
```

```
}while (i < 5);
```

do-while and switch demonstration:

ProgramLink(Sapphire Engine): https://sapphireengine.com/@/z5jrrd

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

NESTED LOOPS

Nested loop means a loop inside another loop. That is why nested loops are also called as “loop inside loop“.

Syntax for Nested For loop:

```
for ( initialization; condition; increment ) {
   for ( initialization; condition; increment ) {
      // statement of inner loop
}
   // statement of outer loop
}
```

Syntax for Nested While loop:

```
while(condition) {
   while(condition) {
      // statement of inner loop
```

```
    }
    // statement of outer loop
}
```

Syntax for Nested Do-While loop:

```
do{
    do{
        // statement of inside loop
    }while(condition);
    // statement of outer loop
}while(condition);
```

Note: There is no rule that a loop must be nested inside its own type. In fact, there can be any type of loop nested inside any type and to any level.

Some implementations of loops:

Program1:

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

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

```
/* Printing the pattern:
    *
    **
    ***
    ****
    *****
*/
#include <iostream>
using namespace std;

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

  for (int i=1; i<=n; i++) {
     for (int j=1; j<=i; j++)
        cout << "*";
     cout << endl;
  }
  return 0;
}
```

Standard Output

```
Enter n: 8
*
**
***
****
*****
******
*******
********
```

Program2:

ProgramLink(Sapphire Engine): https://sapphireengine.com/@/uc300p

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

```
/*  Printing the pattern:
    A
    BA
    CBA
    DCBA
    EDCBA
*/
#include <iostream>
using namespace std;

 int main() {

    int n;
    cout << "Enter n: ";
    cin >> n;
```

```
16.      char ch = 'A';
17.      for (int i=1; i<=n; i++) {
18.          for (int j=1; j<=i; j++)
19.              cout << (char)(ch + i - j) ;
20.          cout << endl;
21.      }
22.      return 0;
23.  }
```

Standard Output

```
Enter n: 5
A
BA
CBA
DCBA
EDCBA
```

Break and continue statements

Break

Just like we use the break statement to jump out of a switch statement, we can use the break statement to jump out of a loop.This will be more clear by the following example.

```
for (int i = 0; i < 10; i++) {

  if (i == 4) {

    break;

  }

  cout << i << "\n";

}
```

Output:

0

1

2

3

Note that when the value of i becomes 4, the control will be shifted out of the loop. Loop will not run for the remaining values of i.

Program3:

ProgramLink(Sapphire Engine):https://sapphireengine.com/@/b7n7dn

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

```
1. #include <iostream>
2. using namespace std;
3.
4. int main()
5. {
6.       int a=0, i;
7.     for (i=0; i<5; i++) {
8.         a++;
9.         if (i == 3)
10.               break;
11.      }
12.      cout << a << endl;
13.     return 0;
14.  }
```

Standard Output

4

Continue

The continue statement breaks one iteration. If a specified condition occurs, then it continues with the next iteration in the loop.

For example,

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  cout << i << "\n";
}
```

In the above example, the value of 4 is skipped. The output will be:

```
0
1
2
3
5
6
7
8
9
```

Program4:

ProgramLink(Sapphire Engine):https://sapphireengine.com/@/hmpnp3

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

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

int main() {

    for (int i=1; i<=3; i++) {
        for (int j=1; j<=3; j++) {
            if(i == j)
                continue;
            for (int k=1; k<=3; k++) {
                if(k == j || k == i)
                    continue;
                cout << i << " " << j << " " << k << endl;
            }
        }
    }
```

Standard Output

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

Question Bank :

1 What is the output?

```
int i = 1;
while(i < 5) {
   if(i == 3) {
      break;
   }
   cout << i << " ";
   i++;

}
```

The output of the above code snippet will be **1 2 .**

The while loop will execute till the value of i is 2. The moment at which the value of variable i becomes 3, the expression in the if statement will evaluate to true and the **break** statement will get executed. Break statement is used to exit from the current loop. The control will be shifted outside the while loop.

2 What is the output?

```
  int main() {

   for(int i = 0; i < 3; i++) {

      cout << i << " ";

   }

   cout << i << " ";

}
```

The above code snippet will give an error. If we look closely variable i is declared inside the for loop and we are printing it's value outside the for loop. So it gives an error because we are printing the value of i ,outside it's scope.

3 What is the output?

```
int main()

{

 for(; ;);

for(; ;);

   cout<<"Hello"<<endl;

return 0;

}
```

blank for loop with ; ; is always infinite loop. printf() will never be executed in this program.

Extra question 1:

https://codeforces.com/problemset/problem/266/B

In this question we are given that there is a queue of n number of people. If at time x a boy is standing on ith position and a girl is standing on (i+1)th position then the boy feels awkward and let's the girl on his position. It means at time x+1, girl will be at ith position and the boy will be at (i+1)th position. We have to print the queue after t seconds.

We can iterate over the whole loop every time to check for all the positions where a boy is standing immediately before a girl. If we encounter such a position then we will swap them. We will do this for each second. Then after t seconds we can display the modified queue.

Program link:https://sapphireengine.com/@/gme95u

Extra question 2:

Given a number n, print all primes smaller than or equal to n.

We know that a prime number has only 2 divisors i.e 1 and the number itself. We can check for each number less than n if it is a prime or not. Now how will we check if it is a prime or not? We will take help of a loop.Suppose we have to check whether num is a prime or not. Using a loop we will iterate from 2 to num-1. If num is divided by any value between 2 and num-1, it means num is not a prime number. Otherwise it is. This approach will require two loops.

Program link for the above approach is:

https://sapphireengine.com/@/3fd4el

https://ideone.com/yRFWoU

```
#include <bits/stdc++.h>
#include <iostream>
using namespace std;

int main()
{
int n;
cin>>n;//1<=n
int flag=0;
 for(int i=2;i<=n;i++)// 1 is not a prime number
 {
    flag=0;
  for(int j=2;j<i;j++)
  {
    if(i%j==0)
    {
      flag=1;
      break;
    }

  }
```

```
22.     if(flag==0)
23.      {
24.          cout<<i<<" ";
25.      }
26.  }
27.           return 0;
28. }
```

But we can do a bit better than this. **The sieve of Eratosthenes** is one of the most efficient ways to find all primes smaller than n when n is smaller than 10 million or so.

The idea is

1. Create a list of consecutive integers from 2 to *n. Suppose n=30*

   2, 3, 4, 5, 6 ,7 ,8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, and 30.

2. *Initially, let p be a variable equal to 2, the first prime number. Mark bold all the multiples of 2 except 2.*

   *2, 3, **4**, 5, **6** ,7 ,**8**, 9, **10**, 11, **12**, 13, **14**, 15, **16**, 17, **18**, 19, **20**, 21, **22**, 23, **24**, 25, √ **26**, 27, **28**, 29, and **30**.*

3. *Increment p. Now p=3. Mark its multiples as bold. Basically mark all the multiples of p as bold (except p) as they are composite numbers.*

   *2, 3, **4**, 5, **6** ,7 ,**8**, **9**, **10**, 11, **12**, 13, **14**, **15**, **16**, 17, **18**, 19, **20**, **21**, **22**, 23, **24**, 25, **26**, **27**, **28**, 29, and **30**.*

4. *Assign the value of the next prime number i.e the next number which is not in bold letters to p. Repeat the process till Repeat the process until p<=$\sqrt{n}$. So do the same for 5. Stop at n=7 as 7> $\sqrt{30}$*

   *2, 3, **4**, 5, **6** ,7 ,**8**, **9**, **10**, 11, **12**, 13, **14**, **15**, **16**, 17, **18**, 19, **20**, **21**, **22**, 23, **24**, **25**, **26**, **27**, **28**, 29, and **30***

   The numbers which are not marked as bold are prime numbers less than 30.

   NOTE: Here we have discussed the basic idea of finding the prime numbers by sieve method. We are not going to implement it now as the implementation requires knowledge of arrays. So we will cover the implementation part in Arrays topic.

Practice Questions:

Here are some practice questions you should try. We recommend you to try these by yourself before looking for the code solution. However their links will be made available to you if you need them.

1 https://codeforces.com/problemset/problem/266/A

2 https://codeforces.com/problemset/problem/1328/A

3 Given a number N, find its square root. You need to find and print only the integral part of the square root of N. You have to find the square root using loops.
Example for N=30 the answer will be 5.

4 Given a decimal number, convert it into binary and print it. Example: decimal no 7 it's decimal equivalent is 0111.

5 https://codeforces.com/contest/263/problem/A

[5th question can be solved without using 2d arrays. Try solving this with loops only]

6 **https://www.codechef.com/problems/PLMU**