1.

```
public static int sum(int n){
      int partialSum = 0;
      for (int i = 0; i < n; i++) {
            partialSum = partialSum + i*i*i;
      }
      return partialSum;
}
```

2.

```
public static long factorial(int n) {
      if (n <=1) {
            return 1;
      } else {
            return n * factorial (n-1);
      }
}
```

3.

```
public static int sum(int n){
      int sum = 0;
      for (int i = 0; i < n; i++) {
            sum = sum + 1;
      }
      return sum;
}
```

4.

```
public static int sum(int n) {
      int sum = 0;
      for (int i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                  sum = sum + 1;
            }
      }
      return sum;
}
```

5.

```
public static int sum(int n) {
      int sum = 0;
      for (int i = 0; i < n; i++) {
            for (int j = 0; j < i * i; j++) {
                  sum = sum + 1;
            }
      }
      return sum;
}
```

6.
```
public static int sum(int n){
      int sum = 0;
      for (int i = 0; i < n; i++) {
            for (int j = 0; j < i; j++) {
                  sum = sum + 1;
            }
      }
      return sum;
}
```

7.
```
public static int sum(int n) {
      int sum = 0;
      for (int i = 0; i < n; i++) {
            for (int j = 0; j < i*i; j++) {
                  for (int k = 0; k <j; k++) {
                        sum = sum + 1;
                  }
            }
      }
      return sum;
}
```

8.
```
//matrix multiplication, we multiply a x*y matrix with a y*z matrix
for (int i = 0; i < x; i++) {
      for (int j = 0; j < y; j++){
            C[i][j] = 0;
            for (int k = 0; k < z; k++) {
                  C[i][j] = C[i][j] + A[i][k] * B[k][j]
            }
      }
}
```

9.
```
public static int mistery(int n) {
      int r = 0;
      for (int i = 1; i <=n-1; i++) {
            for (int j =i+1; j <=n; j++) {
                  for (int k = 1; k <=j; k++) {
                        r = r + 1;
                  }
            }
      }
      return r;
}
```

10.
```
public static int mistery(int n) {
      int r = 0;
      for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                  for (int k = j; k <= j+i; k++) {
                        r = r + 1;
                  }
            }
      }
      return r;
}
```

11.
```
public static int mistery(int n) {
      int r = 0;
      for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++ ){
                  for (int k =j; k <= i+j; k++) {
                        for (int l=1; l <= i+j-k; l++) {
                              r = r + 1;
                        }
                  }
            }
      }
      return r;
}
```

12.
```
public static int mistery(int n) {
      int r = 0;
      for (int i = 1; i <= n; i++) {
            for (int j = i+1; j <=n; j++) {
                  for (int k = i+j – 1; k <=n; k++) {
                        r = r + 1;
                  }
            }
      }
      return r;
}
```

13.
```
public static void function(int n) {
      count = 0;
      for (i = 1; i*i <= n; i++) {
            count = count + 1;
      }
      return count;
}
```

14.
```
public static void function(int n) {
      int i = 1;
      int s = 1;
      while (s <= n) {
            i = i + 1;
            s = s + 1;
      }
      return s;
}
```

15.
```
public static void function(int n) {
      for (int i = 1; i <=n; i++ ) {
            for (int j = 1; j <= n; j = j + i) {
                  System.out.println("*");
            }
      }
}
```

16.
```
public static void function(int n) {
      for (int i = 1; i <= n /3; i++) {
            for (int j = 1; j <= n; j = j + 4) {
                  System.out.println("*");
            }
      }
}
```

17.
```
public static void s1(n){
      for (int i = 1; i <= n; i++) {
            int j = n;
            while (j > 0) {
                  j = j/2;
                  System.out.println("*");
            }
      }
}
```

18.
```
public static void s2(n){
      for (int i =1; i<= n; i++){
            int j = i;
            while (j > 0){
                  j = j/2;
                  System.out.println("*");
            }
      }
}
```

19.
```
public static void s3(x, n, a){
      boolean  found = false;
      for (int i = 0; i < n; i++){
            if (x[i] == a){
                  found = true;
            }
      }
}
```

20.
```
public static void s4(x, n, a){
      boolean found = false;
      while (found == false && i < n) {
            if (x[i] == a) {
                  found = true;
            }
            i = i + 1;
      }
}
```

21.
```
public static void s7(n){
      int s = 0;
      for (int i = 1; i < n *n; i++){
            int j = i;
            while (j > 0) {
                  s = s + j
                  j = j - 1
            }
      }
}
```

22.
```
public static void s8(n){
      int s = 0;
      for (int i = 1; i < n*n; i++){
            int j = i;
            while (j > 0) {
                  s = s + j;
                  j =j / 10;
            }
      }
}
```

23.
```
public static void operation(n, i){
      if (n > 1) {
            m ← n/2;
            operation(m, i-2);
            operation(m, i-1);
            operation(m, i+2);
            operation(m, i+1);
      } else {
            System.out.println(i);
      }
}
```

24.
```
public static int recursiveFun1(int n)
{
    if (n <= 0)
        return 1;
    else
        return 1 + recursiveFun1(n-1);
}
```

25.
```
public static int recursiveFun2(int n)
{
    if (n <= 0)
        return 1;
    else
        return 1 + recursiveFun2(n-5);
}
```

26.
```
public static int recursiveFun3(int n)
{
    if (n <= 0)
        return 1;
    else
        return 1 + recursiveFun3(n/5);
}
```

27.
```
public static void recursiveFun4(int n, int m, int o)
{
    if (n <= 0)
    {
        printf("%d, %d\n",m, o);
    }
    else
    {
        recursiveFun4(n-1, m+1, o);
        recursiveFun4(n-1, m, o+1);
    }
}
```

28.

```
public static int recursiveFun5(int n)
{
    for (i = 0; i < n; i += 2) {
        // do something
    }

    if (n <= 0)
        return 1;
    else
        return 1 + recursiveFun5(n-5);
}
```