Question 1

What will happen if you compile/run this code?

```
1:     public class Q1 extends Thread
2:     {
3:         public void run()
4:         {
5:            System.out.println("Before start method");
6:            this.stop();
7:            System.out.println("After stop method");
8:        }
9:
10:        public static void main(String[] args)
11:        {
12:            Q1 a = new Q1();
13:            a.start();
14:        }
15:    }
```

A) Compilation error at line 7.
B) Runtime exception at line 7.
C) Prints "Before start method" and "After stop method".
D) Prints "Before start method" only.

Answer

Question 2

What will happen if you compile/run the following code?

```
1:    class Test
2:    {
3:      static void show()
4:      {
5:              System.out.println("Show method in Test class");
6:      }
7:}
8:
9:    public class Q2 extends Test
10:   {
11:     static void show()
12:     {
13:             System.out.println("Show method in Q2 class");
14:     }
15:     public static void main(String[] args)
16:     {
17:             Test t = new Test();
18:             t.show();
19:             Q2 q = new Q2();
20:             q.show();
21:
22:             t = q;
23:             t.show();
24:
25:             q = t;
26:             q.show();
27:     }
28: }
```

A) prints "Show method in Test class"

```
          "Show method in Q2 class"
          "Show method in Q2 class"
          "Show method in Q2 class"

B) prints "Show method in Test class"
          "Show method in Q2 class"
          "Show method in Test class"
          "Show method in Test class"

C) prints "Show method in Test class"
          "Show method in Q2 class"
          "Show method in Test class"
          "Show method in Q2 class"

D) Compilation error.
```

[Answer]

Question 3

The following code will give

```
1:    class Test
2:    {
3:      void show()
4:      {
5:              System.out.println("non-static method in Test");
6:      }
7:    }
8:    public class Q3 extends Test
9:    {
10:     static void show()
11:     {
12:             System.out.println("Overridden non-static method in Q3");
13:     }
14:
15:       public static void main(String[] args)
16:       {
17:              Q3 a = new Q3();
18:        }
19:    }
```

A) Compilation error at line 3.
B) Compilation error at line 10.
C) No compilation error, but runtime exception at line 3.
D) No compilation error, but runtime exception at line 10.

[Answer]

Question No :4

The following code will give

```
1:    class Test
2:    {
3:      static void show()
4:      {
5:              System.out.println("Static method in Test");
6:      }
7:    }
```

```
8:    public class Q4 extends Test
9:    {
10:     void show()
11:     {
12:             System.out.println("Overridden static method in Q4");
13:     }
14:     public static void main(String[] args)
15:     {
16:     }
17:    }
```

```
A) Compilation error at line 3.
B) Compilation error at line 10.
C) No compilation error, but runtime exception at line 3.
D) No compilation error, but runtime exception at line 10.
```

Answer

Question No :5

The following code will print

```
int i = 1;
i <<= 31;
i >>= 31;
i >>= 1;

int j = 1;
j <<= 31;
j >>= 31;

 System.out.println("i = " +i );
 System.out.println("j = " +j);
```

```
A) i = 1
   j = 1

B) i = -1
   j = 1

C) i = 1
   j = -1

D) i = -1
   j = -1
```

Answer

Question No :6

The following code will print

```
Double a = new Double(Double.NaN);
Double b = new Double(Double.NaN);

if( Double.NaN == Double.NaN )
  System.out.println("True");
else
  System.out.println("False");

if( a.equals(b) )
  System.out.println("True");
```

```
else
  System.out.println("False");
```

```
A) True
   True

B) True
   False

C) False
   True

D) False
   False
```

Answer

Question No :7

```
if( new Boolean("true") == new Boolean("true"))
    System.out.println("True");
else
    System.out.println("False");
```

```
A) Compilation error.
B) No compilation error, but runtime exception.
C) Prints "True".
D) Prints "False".
```

Answer

Question No :8

```
public class Q8
{
    int i = 20;
    static
    {
        int i = 10;

    }
    public static void main(String[] args)
     {
          Q8 a = new Q8();
          System.out.println(a.i);
     }
 }
```

```
A) Compilation error, variable "i" declared twice.
B) Compilation error, static initializers for initialization purpose only.
C) Prints 10.
D) Prints 20.
```

Answer

Question No :9

```
The following code will give

     Byte b1 = new Byte("127");

     if(b1.toString() == b1.toString())
        System.out.println("True");
    else
        System.out.println("False");

A) Compilation error, toString() is not avialable for Byte.
B) Prints "True".
C) Prints "False".
```

Answer

Question No :10

```
What will happen if you compile/run this code?

1:    public class Q10
2:    {
3:      public static void main(String[] args)
4:      {
5:              int i = 10;
6:              int j = 10;
7:              boolean b = false;
8:
9:              if( b = i == j)
10:                System.out.println("True");
11:             else
12:                System.out.println("False");
13:     }
14:    }

A) Compilation error at line 9 .
B) Runtime error exception at line 9.
C) Prints "True".
D) Prints "Fasle".
```

Answer

Question No: 1

```
    D.  After the execution of stop() method, thread won't execute any more stateme
```

Back to Question 1

Question No: 2

```
    D. Explicit casting is required at line 25.
```

Back to Question 2

Question No: 3

```
    B. You cann't override an non-static method with static method.
```

Back to Question 3

Question No: 4

```
    B. You cann't override a static method with non-static method.
```

Back to Question 4

Question No: 5

```
    D.
```

Back to Question 5

Question No: 6

```
    C.
```

Back to Question 6

Question No: 7

```
    D.
```

Back to Question 7

Question No: 8

```
    D.  Here the variable '"i" defined in static initializer is local to that block
        The statements in the static initializers will be executed (only once) when
```

Back to Question 8

Question No: 9

```
    C.
```

Back to Question 9

Question No: 10

```
    C. Conditional operators have high precedence than assignment operators.
```

Back to Question 10