(1) Write a program to demonstrate Tightly Coupled code.

//Sort and Search Program

```
package com.bootcamp.spring.firstspring.TightlyCoupledQ1;

public class BinarySearchTightlyCoupled {

            public int BinarySearch(int[] arr,int key) {
            //Bubble Sort
             int n = arr.length;
     int temp = 0;
      for(int i=0; i < n; i++){
              for(int j=1; j < (n-i); j++){
                       if(arr[j-1] > arr[j]){
                              //swap elements
                              temp = arr[j-1];
                              arr[j-1] = arr[j];
                              arr[j] = temp;
                       }
                                   }
                       }
      //Binary Search
      int first=0;
      int last=arr.length-1;

      int mid = (first + last)/2;
      while( first <= last ){
         if ( arr[mid] < key ){
           first = mid + 1;
         }else if ( arr[mid] == key ){
//           System.out.println("Element is found at index: " + mid);
           return mid;

         }else{
            last = mid - 1;
         }
         mid = (first + last)/2;

         if ( first > last ){
             System.out.println("Element is not found!");

          }
```

```
  }

  return -1;
    }
}

// Main Program

package com.bootcamp.spring.firstspring.TightlyCoupledQ1;
public class Program {

    public static void main(String[] args) {
        BinarySearchTightlyCoupled obj=new BinarySearchTightlyCoupled();
        int arr[]= {4,3,5,7,3,1,9};
        int num=7;

        int res=obj.BinarySearch(arr, num);
        if(res==-1) {
            System.out.println("Element not found");

        }
        else {
        System.out.println("Element found at index"+res);
    }
    }


}
```

(2) Write a program to demonstrate Loosely Coupled code.

```
//Binary Search
package com.bootcamp.spring.firstspring.LooselyCoupled;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class BinarySearch {

        @Autowired
        private Sorting sorting;
        //Sorting Called dynamically
        BinarySearch(Sorting sorting){
                this.sorting=sorting;
        }

        public int binarySearch(int[] arr,int key) {

                arr=sorting.sort(arr);

                //Searching
                int n = arr.length;
           int temp = 0;
            for(int i=0; i < n; i++){
                 for(int j=1; j < (n-i); j++){
                       if(arr[j-1] > arr[j]){
                           //swap elements
                           temp = arr[j-1];
                           arr[j-1] = arr[j];
                           arr[j] = temp;
                       }
                               }
                       }
            //Binary Search
            int first=0;
            int last=arr.length-1;

            int mid = (first + last)/2;
            while( first <= last ){
              if ( arr[mid] < key ){
                first = mid + 1;
              }else if ( arr[mid] == key ){
```

```
        return mid;

       }else{
        last = mid - 1;
       }
       mid = (first + last)/2;

       if ( first > last ){
         return -1;

        }
      }

      return -1;


    }
}

//QUICK SORT
package com.bootcamp.spring.firstspring.LooselyCoupled;

import org.springframework.stereotype.Component;

@Component
public class QuickSort implements Sorting{

    @Override
    public int[] sort(int[] arr) {
        int low = 0 , high = arr.length-1;
        int pivot = arr[high];
        int i = (low-1);
        for (int j=low; j<high; j++)
        {
            if (arr[j] <= pivot)
            {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i+1];
        arr[i+1] = arr[high];
```

```
            arr[high] = temp;
            return arr;
      }
}

//SELECTION SORT
package com.bootcamp.spring.firstspring.LooselyCoupled;

import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;
@Primary
@Component
public class SelectionSort implements Sorting{

      @Override
      public int[] sort(int[] arr) {
          for (int i = 0; i < arr.length - 1; i++)
           {

             int index = i;
             for (int j = i + 1; j < arr.length; j++){
                 if (arr[j] < arr[index]){
                     index = j;
                 }
             }
             int smallerNumber = arr[index];
             arr[index] = arr[i];
             arr[i] = smallerNumber;
           }
          return arr;
      }

}

//SORTING INTERFACE

package com.bootcamp.spring.firstspring.LooselyCoupled;

public interface Sorting
{
int[] sort(int[] arr);
}
```

(3) Use @Compenent and @Autowired annotations to in Loosely Coupled code for dependency management

```
//SelectionSort Class
package com.bootcamp.spring.firstspring.LooselyCoupled;
import org.springframework.stereotype.Component;

@Component
public class SelectionSort implements Sorting{

        @Override
        public int[] sort(int[] arr) {
            for (int i = 0; i < arr.length - 1; i++)
             {

                int index = i;
                for (int j = i + 1; j < arr.length; j++){
                  if (arr[j] < arr[index]){
                    index = j;
                  }
                }
                int smallerNumber = arr[index];
                arr[index] = arr[i];
                arr[i] = smallerNumber;
             }
            return arr;
        }

}
```

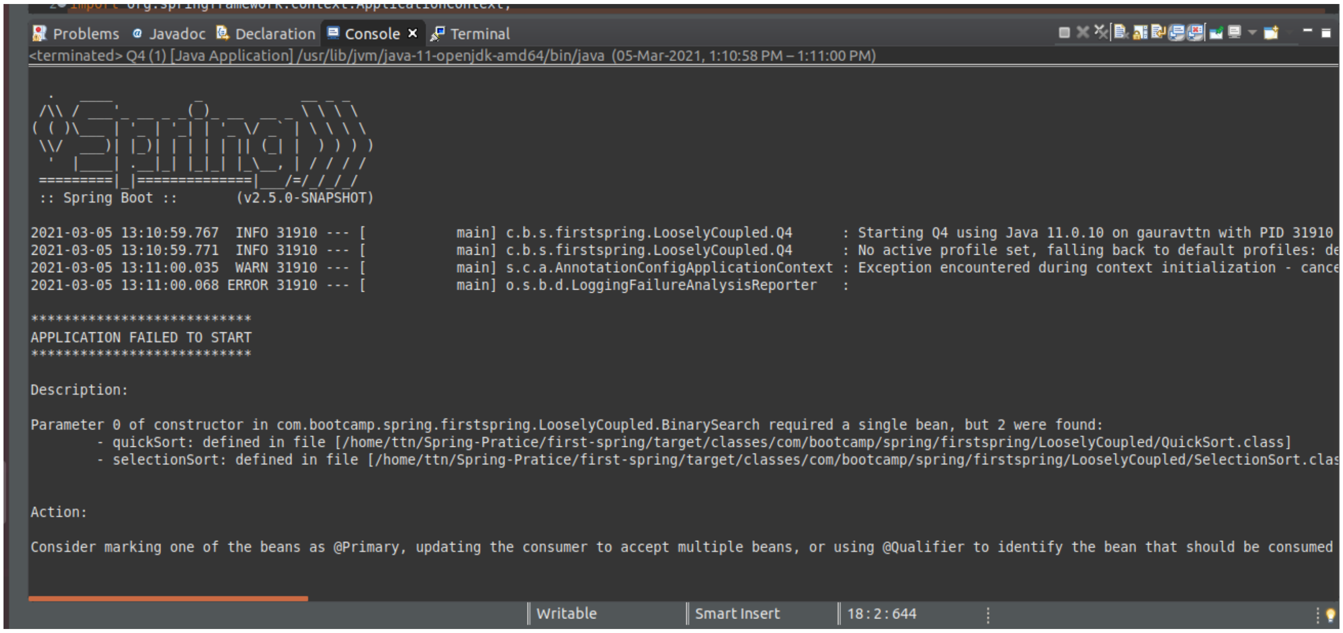
```
//BinarySearch Class
package com.bootcamp.spring.firstspring.LooselyCoupled;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class BinarySearch {

        @Autowired
        private Sorting sorting;
        //Sorting Called dynamically
        BinarySearch(Sorting sorting){
                this.sorting=sorting;
        }

        public int binarySearch(int[] arr,int key) {

                arr=sorting.sort(arr);
                        //Searching
                int n = arr.length;
           int temp = 0;
            for(int i=0; i < n; i++){
                 for(int j=1; j < (n-i); j++){
                       if(arr[j-1] > arr[j]){
                            //swap elements
                            temp = arr[j-1];
                            arr[j-1] = arr[j];
                            arr[j] = temp;
                       }
                                }
                        }
            //Binary Search
            int first=0;
            int last=arr.length-1;

            int mid = (first + last)/2;
            while( first <= last ){
              if ( arr[mid] < key ){
               first = mid + 1;
              }else if ( arr[mid] == key ){
               return mid;
              }else{
                last = mid - 1;
              }
```

```
       mid = (first + last)/2;
       if ( first > last ){
          return -1;
        }
      }
      return -1;
    }
}
```

```
package com.bootcamp.spring.firstspring.LooselyCoupled;

import org.springframework.stereotype.Component;

@Component
public class QuickSort implements Sorting{

    @Override
    public int[] sort(int[] arr) {
        int low = 0 , high = arr.length-1;
        int pivot = arr[high];
        int i = (low-1);
        for (int j=low; j<high; j++)
        {
            if (arr[j] <= pivot)
            {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i+1];
        arr[i+1] = arr[high];
        arr[high] = temp;
        return arr;
    }

}
```

(4) Get a Spring Bean from the application context and display its properties.

```
package com.bootcamp.spring.firstspring.LooselyCoupled;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

import org.springframework.boot.SpringApplication;
@Configuration
@ComponentScan
public class Q4 {
public static void main(String[] args) {

        ApplicationContext app = SpringApplication.run(Q4.class,args);
        BinarySearch searching = app.getBean(BinarySearch.class);

        int[] numbers = {5,55,67,89,71,101,2,88,110,56};
        System.out.println("Index of given Value is - " +

        searching.binarySearch(numbers, 110));
        }
}
```

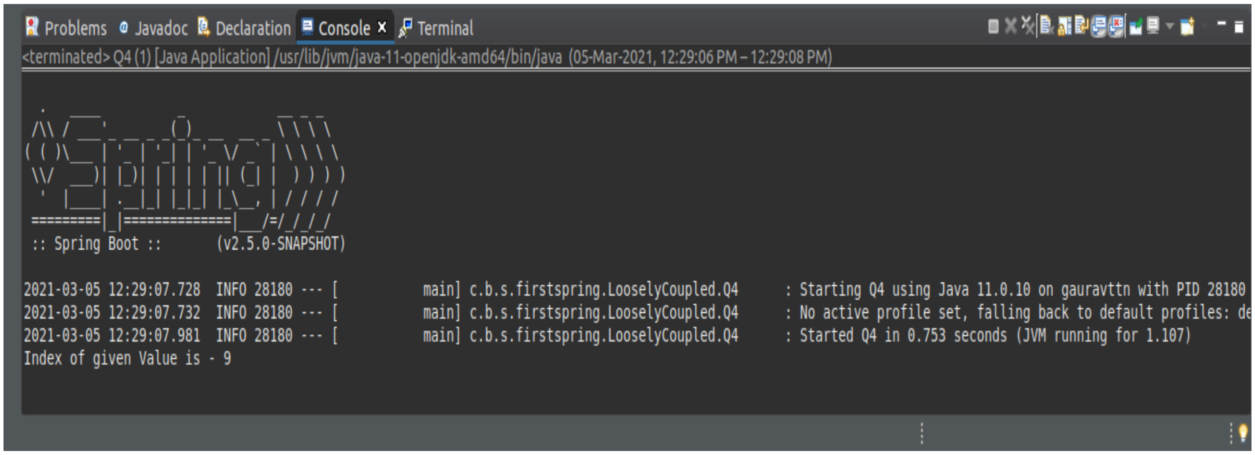


```
<terminated> Q4 (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java  (05-Mar-2021, 1:10:58 PM – 1:11:00 PM)

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::       (v2.5.0-SNAPSHOT)

2021-03-05 13:10:59.767  INFO 31910 --- [           main] c.b.s.firstspring.LooselyCoupled.Q4      : Starting Q4 using Java 11.0.10 on gauravttn with PID 31910
2021-03-05 13:10:59.771  INFO 31910 --- [           main] c.b.s.firstspring.LooselyCoupled.Q4      : No active profile set, falling back to default profiles: de
2021-03-05 13:11:00.035  WARN 31910 --- [           main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cance
2021-03-05 13:11:00.068 ERROR 31910 --- [           main] o.s.b.d.LoggingFailureAnalysisReporter   :

***************************
APPLICATION FAILED TO START
***************************

Description:

Parameter 0 of constructor in com.bootcamp.spring.firstspring.LooselyCoupled.BinarySearch required a single bean, but 2 were found:
	- quickSort: defined in file [/home/ttn/Spring-Pratice/first-spring/target/classes/com/bootcamp/spring/firstspring/LooselyCoupled/QuickSort.class]
	- selectionSort: defined in file [/home/ttn/Spring-Pratice/first-spring/target/classes/com/bootcamp/spring/firstspring/LooselyCoupled/SelectionSort.clas


Action:

Consider marking one of the beans as @Primary, updating the consumer to accept multiple beans, or using @Qualifier to identify the bean that should be consumed
```

(5) Demonstrate how you will resolve ambiguity while autowiring bean (Hint : @Primary)

```
package com.bootcamp.spring.firstspring.LooselyCoupled;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;
@Primary                    //We use @Primary to resolve ambiguity
@Component
public class SelectionSort implements Sorting{

        @Override
        public int[] sort(int[] arr) {
            for (int i = 0; i < arr.length - 1; i++)
             {
                int index = i;
                for (int j = i + 1; j < arr.length; j++){
                    if (arr[j] < arr[index]){
                        index = j;
                    }
                }
                int smallerNumber = arr[index];
                arr[index] = arr[i];
                arr[i] = smallerNumber;
             }
            return arr;
        }}
```

//After @Adding Primary to the Selection Sort Class



```
<terminated> Q4 (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (05-Mar-2021, 12:29:06 PM – 12:29:08 PM)

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::       (v2.5.0-SNAPSHOT)

2021-03-05 12:29:07.728  INFO 28180 --- [           main] c.b.s.firstspring.LooselyCoupled.Q4      : Starting Q4 using Java 11.0.10 on gauravttn with PID 28180
2021-03-05 12:29:07.732  INFO 28180 --- [           main] c.b.s.firstspring.LooselyCoupled.Q4      : No active profile set, falling back to default profiles: de
2021-03-05 12:29:07.981  INFO 28180 --- [           main] c.b.s.firstspring.LooselyCoupled.Q4      : Started Q4 in 0.753 seconds (JVM running for 1.107)
Index of given Value is - 9
```

(6) Perform Constructor Injection in a Spring Bean

```
package com.bootcamp.spring.firstspring.LooselyCoupled;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class BinarySearch {

    @Autowired
    private Sorting sorting;
    //Sorting Called dynamically

    //**********Constructor Injection***********
    BinarySearch(Sorting sorting){
        this.sorting=sorting;
    }

    public int binarySearch(int[] arr,int key) {

        arr=sorting.sort(arr);

        //Searching
        int n = arr.length;
      int temp = 0;
      for(int i=0; i < n; i++){
         for(int j=1; j < (n-i); j++){
            if(arr[j-1] > arr[j]){
              //swap elements
              temp = arr[j-1];
              arr[j-1] = arr[j];
              arr[j] = temp;
            }
                }
            }
      //Binary Search
```

```
        int first=0;
        int last=arr.length-1;

        int mid = (first + last)/2;
        while( first <= last ){
          if ( arr[mid] < key ){
            first = mid + 1;
          }else if ( arr[mid] == key ){
            return mid;

          }else{
             last = mid - 1;
          }
          mid = (first + last)/2;

          if ( first > last ){
              return -1;

           }
        }

        return -1;


    }
}
```