Question 1

Bed consists of bed frame and mattress. Given Program B1 as shown below. Answer the following questions (a) and (b).

```
1  // Program B1
2  class BedFrame
3  {     private String made;
4
5        public BedFrame(String made)
6        {     this.made = made; }
7
8        public String getMade()
9        {     return made; }
10 }
11
12 class Mattress
13 {     private String size;       private String material;
14       private String color;
15
16       public Mattress(String size, String material, String color)
17       {     this.size = size;
18             this.material = material;
19             this.color = color; }
20
21   public String getMattress()
22       {     return size + '\n' + material + ", " + color; }
23 }
24
25 class Bed
26 {     private BedFrame bedFrame; private String type;
27       private Mattress mattress;
28
29   public Bed(BedFrame frame, Mattress mattress, String type)
30       {     bedframe = frame;
31             this.mattress = mattress;  this.type = type;}
32
33       public String getType()
34       {     return type; }
35       public String toString()
36       { return '\n' + getType() + '\n' + bedFrame.getMade() +
37                 '\n' + mattress.getMattress() +'\n'; }
38 }
39
40 public class TestBed1 {
41   public static void main(String args[]){
42     BedFrame bedFrame1 = _____________________________;
43     Mattress mattress1 = _____________________________;
44     Bed bed1 = _____________________________;
45     System.out.println (bed1.toString());
46     BedFrame bedFrame2 = _____________________;
47     Mattress mattress2 =_____________________________;
48     Bed bed2 = _____________________________________;
49     System.out.println (bed2.toString());
50   }
51 }
```

Figure B1

a. Draw a UML class diagram that shows the relationship among classes in Program B1.

b. Based on the output in Figure B2, write codes to create the object(s) for appropriate class with appropriate argument(s) to complete the **`main()`** method in TestBed1 class. Write your answer in Table 1.

```
Fix
Iron
King Size
Foam, White


Sofa
Iron
Single Bed
Spring, Blue
```

Figure B2

Table 1: Answer for Question 1

Line	Java Statement(s)
42	
43	
44	
46	
47	
48	

Answer:

Line	*Java Statement(s)*
36	`new BedFrame("Iron");`
37	`new Mattress ("King Size","Foam","white");`
38	`new Bed (bedFrame1,mattress1,"Fix");`
40	`new bedFrame1 ()`
41	`new Mattress("Single Bed","Spring","Blue");`
42	`new Bed (bedFrame2, mattress2,"Sofa");`

Question 2
Consider the following classes:

```
1   //Program B2
2   public class Transport {
3        public Transport() {
4         System.out.println("Transport Constructor");
5        }
6        public void car1() {
7         System.out.println("Kancil");
8        }
9        public void car2() {
10        System.out.println("Viva");
11       }
12       public String toString() {
13        return ("Alza");
14       }
15   }
16
17  public class Proton extends Transport {
18         public Proton() {
19          System.out.println("Proton Constructor");
20         }
21         public void car1() {
22          System.out.println("Persona");
23         }
24         public void car2() {
25          super.car1();
26         }
27         public String toString() {
28          return super.toString() + " is my favourite car";
29         }
30  }
31
32  public class Vehicle extends Proton{
33          public static void main(String[] args) {
34           Proton mycar = new Proton();
35           Transport yourcar = new Transport();
36           mycar.car1();
37           yourcar.car2 ();
38           mycar.car2();
39           System.out.println(mycar); }
40
41          public Vehicle() {
42           System.out.println ("Vehicle Constructor"); }
43  }
```

Figure B3

a) Write the output from **`Program B2.`**

b) If `toString()` method in `Proton` class is not provided, what will happen if the statement in line 39 is executed?

c) What will be the output if `Vehicle` class in Figure B3 is changed as in Figure B4 below:

```
1   public class Vehicle extends Proton{
2       public static void main(String[] args) {
3
4         new Vehicle ();
5
6            }
7   public Vehicle() {
8           System.out.println ("Vehicle Constructor");
9      }
10  }
```

Figure B4

d) In Java, super class's constructor is not inherited by its subclasses. They are invoked explicitly or implicitly. Briefly explain how to implement both of these invocations based on Program B2 in Figure B3.

Answer:

a. Persona
Viva
Kancil
Alza is my favourite car

b. The output will display hash code / address

c. Transport Constructor
Proton Constructor
Vehicle Constructor

d. Implicit call - When a subclass is instantiated, the superclass default constructor is executed first
Explicit call – Using super keyword
Line 18 – Implicit call

.

Question 3

Given the following program (Program B3), answer questions below:

a) Complete the class declarations with an appropriate Java statement based on the instructions below:

i) Declare class named **ClassA** that implements interface class named **InterfaceA**.

ii) Declare class named **ClassB** that inherits class **ClassA** and implements interface class **InterfaceA**.

iii) Declare class named **ClassC** that inherits class **ClassB**.

b) Assume that the program is successfully completed in (a), determine the output of the program if each of the following Java statements is inserted into the program at **line 53 and 54**. ***Note:*** each question below is independent.

i)
```
ClassA obj = new ClassB("Nice");
obj.print();
```

ii)
```
ClassA obj = new ClassC();
obj.print();
```

```
1  //Program B3
2  interface InterfaceA {
3      public void print();
4  }
5
6  //Declare class named ClassA that implements interface
7  //class named InterfaceA
8  _________________a(i)_____________ {
9      public ClassA() {
10        this("Hello");
11        System.out.println("Default constructor for ClassA");
12     }
13
14     public ClassA(String str) {
15        System.out.println("Constructor with argument for ClassA: " +
16        str);
17     }
18
19     public void print() {
20        System.out.println("Using print method in ClassA");
21     }
22 }
23
24 //Declare class named ClassB that inherits class named //ClassA and
25 implements interface class named InterfaceA
26 _________________a(ii)_____________ {
27     public ClassB() {
28        super("Hi");
29        System.out.println("Default constructor for ClassB");
30     }
31
32     public ClassB(String str) {
33        System.out.println("Constructor with argument for ClassB: " +
34        str);
35     }
36
37     public void print() {
38        System.out.println("Using print method in ClassB");
39     }
40 }
41
42 //Declare class named ClassC that inherits class named ClassB
43 _________________a(iii)_____________ {
44 {
```

```
45      public ClassC() {
46         System.out.println("Default constructor for ClassC");
47      }
48  }
49
50  public class PolyTest
51  {
52      public static void main(String [] args) {
53         _________________(b)_____________
54         ____________________________________
55      }
56  }
```

Answer:

a(i) class ClassA implements InterfaceA
a(ii) class ClassB extends ClassA implements InterfaceA
a(iii) class ClassC extends ClassB

b(i) Constructor with argument for ClassA: Hello
Default constructor for ClassA
Constructor with argument for ClassB: Nice
Using print method in ClassB

b(ii) Constructor with argument for ClassA: Hi
Default constructor for ClassB
Default constructor for ClassC
Using print method in ClassB

Question 4

a. Describe two differences between checked and unchecked exceptions.

b. Change the following method so that it catches the `IllegalArgumentException` thrown by `divide` and prints the message "`the divisor is zero`" if it is thrown.

```
public static void printQuotient(double a, double b) {
  System.out.println(divide(a, b));
}
```

c. Write the output for the following program.

```
1   class Exceptions {
2    public static void main(String[] args) {
3    String languages[] = { "C", "C++", "Java", "Perl", "Python" };
4    try {
5       for (int c = 1; c <= 5; c++) {
6         System.out.println(languages[c]);
7       }
8     }
9     catch (Exception e) {
10      System.out.println(e); }
11    }
12  }
```

d. Given the "**Power3Integer.java**" source file and its input/ output example as follows:

```
1   import java.util.Scanner;
2   import java.util.InputMismatchException;
3
4   class Power3Integer {
5     public static void main(String[] args) {
6       int num = 0;
7       Scanner input = new Scanner(System.in);
8
9       try {
10        System.out.print("Number: ");
11        num = input.nextInt();
12        System.out.println("Result: " + (num * num * num));
13
14      } catch (InputMismatchException ex) {
15        System.out.println("Please enter valid integer number.");
16        input.nextLine();
17      }
18    }
19  }
```

Power3Integer.java (Source file)

```
Number: one
Please enter valid integer number.
```

Power3Integer.java (Input/ output example)

Modify the exception handling control inside the "**Power3Integer.java**" codes to repeatedly asking a new input, if the user enter an invalid integer number. Below is the example of input/ output pattern should look like after the modification was made.

Power3Integer.java (Input/ output example after modification)

```
Number: one
Please enter valid integer number.

Number: two
Please enter valid integer number.

Number: 3
Result: 27
```

Answer:

a) *Checked Exceptions*
- *subclass of Exception*
- *must be handled by programmer (either with try-catch block or declared in method signature as thrown)*

Unchecked Exceptions
- *subclass of RuntimeException*
- *program will compile without explicit handling of unchecked exceptions*

b)
```
public static void printQuotient(double a, double b) {
    try {
        System.out.println(divide(a, b));
    } catch (IllegalArgumentException e) {
        System.out.println("the divisor is zero");
    }
}
```

c) *C++*
Java
Perl
Python
java.lang.ArrayIndexOutOfBoundsException: 5

d)
```
import java.util.Scanner;
import java.util.InputMismatchException;

class Power3Integer {
  public static void main(String[] args) {
    int num = 0;
    Scanner input = new Scanner(System.in);

    boolean repeat = true;

    while (repeat) {
      try {
        System.out.print("Number: ");
        num = input.nextInt();
        System.out.println("Result: " + (num * num * num));
        repeat = false;

      } catch (InputMismatchException ex) {
        System.out.println("Please enter valid integer number.");
        input.nextLine();
      }
    }
  }
}
```