CSE 2001: Data Structure & Algorithms

Programming Assignment-III

(Exception, Generics, & Recursion)

Question-1:

Write a Java program to read your lucky number from keyboard. Treat –ve no. as ***NumberFormatException***. Write appropriate Exceptional handler.

Sample run-1:

```
Enter your lucky number
-90
java.lang.NumberFormatException: Negative number
```

Sample run-2:

```
Enter your lucky number
34
Your lucky number is 34
```

Question-2:

Assign your favourite colours in an array. Identify 2 exceptions that may be generated & write exceptional handler in Java. Also, display the four colours after handling any 2 exceptions.

Sample run:

```
Enter four colours
RED
BLUE
YELLOW
GREEN
Convert string to integer
java.lang.NumberFormatException: For input string: "RED"
Enter one more colour
VIOLET
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of
bounds for length 4
The colours entered are
RED
BLUE
YELLOW
GREEN
```

Question-3:

Create a class Student having two instance variable name and mark. Enter mark, name of the student. If mark is more than 100, create exception ***MarksOutOfBoundException*** & throw it using Java. Display the customised message Mark can't be greater than 100 for the exception.

Sample run-1:

```
Enter the name of the student
RAMESH
Enter marks
98
RAMESH has got 98.0
```

Sample run-2:

```
Enter the name of the student
Raju
Enter marks
130
MarksOutOfBoundException: Mark can't be greater than 100
```

Question-4:

Write a java program to print an array of different type using a single Generic method. The signature of *printArray* method is given below.

```
public static < E > void printArray( E[] inputArray)
```

Sample run:

```
Enter array elements
1
2
3
4
5

Integer Array contains:
1  2  3  4  5

Enter array elements
1.2
2.3
3.4
4.5
5.6

Double Array contains:
1.2  2.3  3.4  4.5  5.6
```

Question-5:

Write a java method using Generics to count the occurrence of an element in an array of any type. The signature of *count* method is given below.

public static <T> int count(T[] array, T item)

Sample run:

```
Enter array elements
1
2
3
4
5
Enter the element to search
4
Number of times 4 present in the array is 1
```

Question-6:

Write a simple main class in Java that contains an experiment that uses the generic Box<T> class to build boxes with different types and that verifies that this class works as advertised. Your experiment should include the following:

- Create a boxed String object and two variables that refer to that box. Change the contents of one and determine the effect on the other.

- Create a boxed Integer object and two variables that refer to that box. Change the contents of one and determine the effect on the other.

- Create a boxed Object object and two variables that refer to that box. Determine what happens if you put a string in the box. Determine what happens if you put an integer in the box.

Question-7:

Write a menu driven program to perform several mathematical operations. Different choices for the mathematical operations are as follows.

1. Determine the factorial of a number
2. Determine X^N for two numbers X, N
3. Determine GCD of two number.
4. Binary equivalent of a decimal number
5. Product of two numbers.

NOTE: All the mathematical operations must be performed using the recursive method.

Question-8:

Write a recursive method in Java which, given an integer n, print it with its digits reversed. For example, given 4735, it prints 5374

Sample run:

```
Enter the number that you want to reverse:
3456
The reverse of the given number is: 6543
```

Question-9:

The sequence of numbers 1, 1, 2, 3, 5, 8, 13 etc are called Fibonacci numbers, each is the sum of the preceding two. Write a recursive method in Java which, given *n*, returns the *nth* Fibonacci number.

Sample run:

```
Enter one number
10
Fibonacci number at position 10 is 55
```

Question-10:

Write a recursive Java method that takes a character string s and outputs its reverse. For example, the reverse of ' *pots&pans*' would be ' *snap&stop*'.

Home Assignment

Question-1:

Create a class Bank with instance variables account_no, name, and balance of the customer. If the input balance is less than or equal to zero then create an Exception called "Invalid BalanceException" and throw it using Java. Display the custom message "Balance cannot be less than 0".

Sample run-1:

```
Enter name
Rahul
Enter Account number
1235
Enter balance
700
Details of the Account Holder
Name: Rahul
Account number: 1235
```

```
Balance: 700.0
Enter the money to withdraw
300
After withdraw, Balance = 400.0
```

Sample run-2:

```
Enter name
Rahul
Enter Account number
1235
Enter balance
700
Details of the Account Holder
Name: Rahul
Account number: 1235
Balance: 700.0
Enter the money to withdraw
900
InvalidBalanceException: Balance cannot be less than 0
```

Question-2:

Write a recursive method in Java to search an element of an array using *binary search*.

Sample run-1:

```
Enter number of elements in the array
5
Enter 5 number of elements in ascending order
11
22
33
44
55
Enter the element to search
33
The 33 is present at index 2
```

Sample run-2:

```
Enter number of elements in the array
5
Enter 5 number of elements in ascending order
11
22
33
44
55
Enter the element to search
66
```

```
The 66 is not present in the array
```

Question-3:

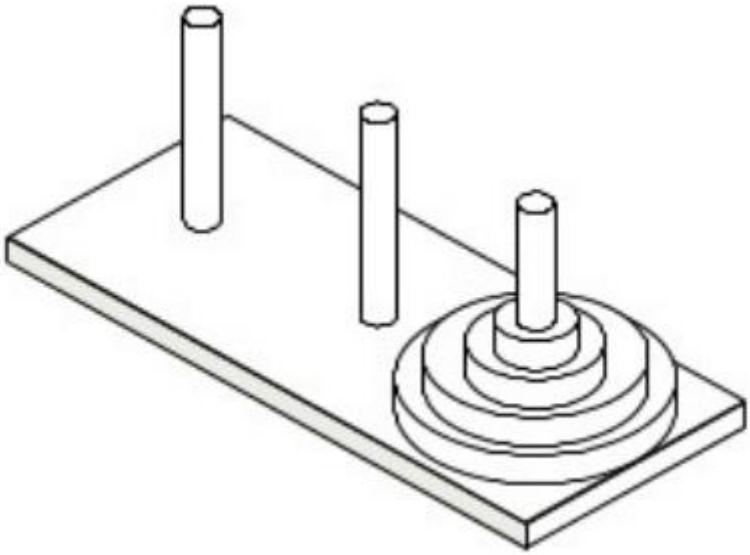
Write a recursive Java method that determines if a string s is a palindrome, that is, it is equal to its reverse.
Examples of palindromes include 'racecar' and 'gohangasalamiimalasagnahog'.

Question-4:

Given an unsorted array, *A*, of integers and an integer *k*, write recursive program using Java for rearranging the elements in *A* so that all elements less than or equal to *k* come before any elements larger than *k*.

Question-5:

In the Towers of Hanoi puzzle, we are given a platform with three pegs, *a*, *b*, and *c*, sticking out of it. On peg *a* is a stack of *n* disks, each larger than the next, so that the smallest is on the top and the largest is on the bottom. The puzzle is to move all the disks from peg *a* to peg *c*, moving one disk at a time, so that we never place a larger disk on top of a smaller one. See Figure 1 for an example of the case $n = 4$. Write a recursive program using Java for solving the Towers of Hanoi puzzle for arbitrary *n*. (Hint: Consider first the sub problem of moving all but the *nth* disk from peg *a* to another peg using the third as "temporary storage.")



****