Please fill in your Student Number and, optionally, Name.	
Student Number	: ____________________
Name	: ____________________

For Official Use	
Mark	: __________
Marker	: __________

University of Cape Town ~ Department of Computer Science

Computer Science 1016S ~ 2007

Theory Test 1A – Supplementary Test

Marks : 30

Time : 40 minutes

Instructions:

a) Answer all questions.
b) Write your answers in the space provided.
c) Show all calculations where applicable.

Question 1: Recursion and Exceptions [15]

```java
import java.util.Scanner;

public class RecursiveTest4 {
   public static void main(String[] args) {
     Scanner keyboard = new Scanner(System.in);
     System.out.print("Enter the height:");
     Tower(keyboard.nextInt(),"");
   }

   public static String Line(int n, char Symbol) {
     if (n==1)
        return(Symbol+"");
     else
        return(Symbol+Line(n-1,Symbol));
   }

   public static void Tri(int n,String offset,char Symbol) {
     System.out.println(offset+Line(n*2+1,Symbol));
     System.out.println(offset+ Symbol + Line(n*2-1,' ') + Symbol);
   }

   public static void Tower(int n, String offset) {
     if(n!=1)
        Tower(n-1, offset + " ");
     Tri(n,offset,'*');
   }
  }
```

For different values of the height parameter, the program listed above produces output as follows.

```
Enter the height:1
***
* *


Enter the height:2
 ***
 * *
*****
*   *
```

```
Enter the height:3
  ***
  * *
 *****
 *   *
*******
*     *
```

etc.

a) What is the stopping case for the `Tower` method?

[1]

(n==1)

b) This program can generate a `java.util.InputMismatchException`. When could this occur?

[1]

____

____

When the user does not enter a an integer [1]

c) Is `java.util.InputMismatchException` is an unchecked exception. Explain what this means.

[2]

____

____

Unchecked exceptions do not have to conform to Java's can-or-declare rule. This means that you do not have to handle or decalre them,

d) Rewrite the main method so that the `java.util.InputMismatchException` is handled.

[3]

```
public static void main(String[] args) {
```

____

____

____

____

____

____

____

____

```
}
```

```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
   try{  // 1 mark
    System.out.print("Enter the height:");
    Tower(keyboard.nextInt(),"");
    }
    catch (InputMismatchException e)  // 1 mark
    {
      System.out.println("You need to type in an integer!");  // 1 mark for sensible message.
    }
  }
```

e) Re –write the method `Tower` so that it produces output as follows:

```
Enter the height:1                           * *
***
```

```
Enter the height:2
*****
*   *
 ***
 * *
```

```
Enter the height:3
*******
*     *
 *****
 *   *
  ***
  * *
```

etc.

[2]

```
public static void Tower(int n, String offset) {
________________________________________________
________________________________________________
________________________________________________
________________________________________________
________________________________________________
}
```

```
public static void Tower(int n, String offset) {
        Tri(n,offset,'*');  //1
         if(n!=1)
           Tower(n-1, offset + " ");  //2
    }
```

// note no marks for an iterative solution!

f) Now, write a recursive definition for the method Tri so that the program produces output as follows:

```
Enter the height:1
*


Enter the height:2
 *
 *
***


Enter the height:3
  *
  *
 ***
  *
 ***
*****
```

```
Enter the height:4
   *
   *
  ***
   *
  ***
 *****
   *
  ***
 *****
*******
```

And so on... You are given the method head, just fill in the body of the method. (Note that no marks will be given for iterative solutions!)

```
public static void Tri(int n,String offset,char Symbol) {










}
```

[6]

```
if (n==1)
        System.out.println(offset+Line(1,Symbol)); //2
     else {
        Tri(n-1,offset+" ",Symbol);  //2
        System.out.println(offset+Line(n*2-1, Symbol)); //2
         }
```

Question 2: File I/O and Exceptions [15]

```
import java.io.*;
import java.util.Scanner;
public class IOTest1 {
  public static void main(String[] args) throws
                                    FileNotFoundException {
    int [] arr = {1,5,8,57,100,119,300,401,1000,12000};
    PrintWriter printW = null;
    printW = new PrintWriter(new FileOutputStream("fileA.txt"));
    Scanner scan = new Scanner(new FileInputStream("fileB.txt"));
    int k = scan.nextInt();
    int p = look(arr,0,10,k);
    printW.println(p);
  }

public static int look(int[] a, int first, int last, int key)  {
    int mid,result=0;
    boolean found = false;
    while ( (first <= last) && !(found) ) {
        mid = (first + last)/2;
        if (key == a[mid]) {
             found = true;
             result = mid;
         }
        else if (key < a[mid]) last = mid - 1;
        else if (key > a[mid]) first = mid + 1;
    }
   if (first > last) result = -1;
   return result;
}
```

a) Is a a `FileNotFoundException` a checked exception? Justify your answer.

[2]

Yes.[1] It is not caught and had to be specified using a throws clause [1]

b) Explain the difference between text files and binary files.

[2]

Text files use the ASCII enconding system and can be read by humans/are portable to any computer.[1] Binary files contain a sequence of bits and are typically meant only to be read by the same programming language on the same computer (Java binary files are portable). [1 mark each for exaplining what the file types are] [2]

c) The program above does not close any of the streams that it opens. Explain clearly why this can be a problem.

[3]

When writing to a file, the write are usually buffered and not perfomed immediately, [1] Closing a file calls the flush() method which flushes the buffer and forces all write to be preformed, [1] If this is not done, Java will close the file, but some data may be missing [1]

d) The method look above returns the position of a key in an array, and -1 if the key is not present. You would prefer to throw an ElementNotPresentException if the key is not present. Write a suitable class defintion for the ElementNotPresentException.

[6]

```
public class ElementNotPresentException extends Exception [2]
{
   public ElementNotPresentException( ) [2]
   {
      super("Element not present in array");
   }
```

```
    public ElementNotPresentException(String message) [2]
    {
      super(message);
    }
}
```

e) Give an example of the use of an **anonymous object** in the program above.

[1]

```
printW = new PrintWriter(new FileOutputStream("fileA.txt"));  <-
FileOutputStream [1]
```

f) Give name of a Dutch Computer Scientist who invented a shortest paths algorithm for weighted graphs.

[1]

Edsgar Dijkstra