

**ECU178 Computer Science:  
210CT - Programming, Algorithms and Data  
Structures Portfolio**

Due on Monday, December 15th, 2014

*Dr James Shuttleworth*

**Robert Rigler : 4939377**

## Contents

<b>Item 1: Week 3 - Linear Search and Duplicate Finder</b>	<b>3</b>
Pre-Homework 1: Write a Program that displays your name 10 times . . . . .	3
Pre-Homework 2: Write a function that draws a square of stars given as a parameter	5
Pre-Homework 3: Write a program to open a file and display it's contents in capitals	7
1. Pseudocode for linear search . . . . .	8
2. Pseudocode for finding duplicates in a list . . . . .	8
<b>Item 2: Week 4 - Time complexities and Big-O notation</b>	<b>9</b>
1. Describe the runtime bounds of the linear search algorithm . . . . .	9
2. Describe the runtime bounds of the duplicate finder algorithm . . . . .	9
Additional work: Critical values of relative runtimes . . . . .	10
<b>Item 3: Week 6 - Harmonic Series or Pivot Selection</b>	<b>12</b>
Pre-Homework 1.: 3-Bit binary number . . . . .	12
Pre-Homework 2.: Boolean sequence binary number . . . . .	12
1. Harmonic Series (Pseudocode) . . . . .	13
2. Harmonic Series (JAVA Implementation) . . . . .	13
<b>Item 4: Week 7 - Heapworksheet or RPN Calculator</b>	<b>15</b>
<b>Item 5: Week 8 - Linked List Delete function or Linked List Sortings</b>	<b>16</b>

## Item 1: Week 3 - Linear Search and Duplicate Finder

### Pre-Homework 1: Write a Program that displays your name 10 times

Listing 1: NameRepeat class JAVA code

```
1  /**
2   * Created by Rob on 23/10/2014.
3   */
4  public class NameRepeat {
5
6      public static void main(String[] args){
7
8          NameRepeat myObject = new NameRepeat(); /*Create Object*/
9
10         /*Use object to call PrintName() method*/
11         myObject.PrintName("Rob");
12     }
13
14     public void PrintName(String _name){
15
16         for (int i = 0; i<10;i++){ /* Loop 10 times*/
17
18             /*, print the number && _name parameter each time.*/
19             System.out.println((i+1) + " " + _name);
20
21         }
22     }
23
24
25 }
26 }
```

### Evidence



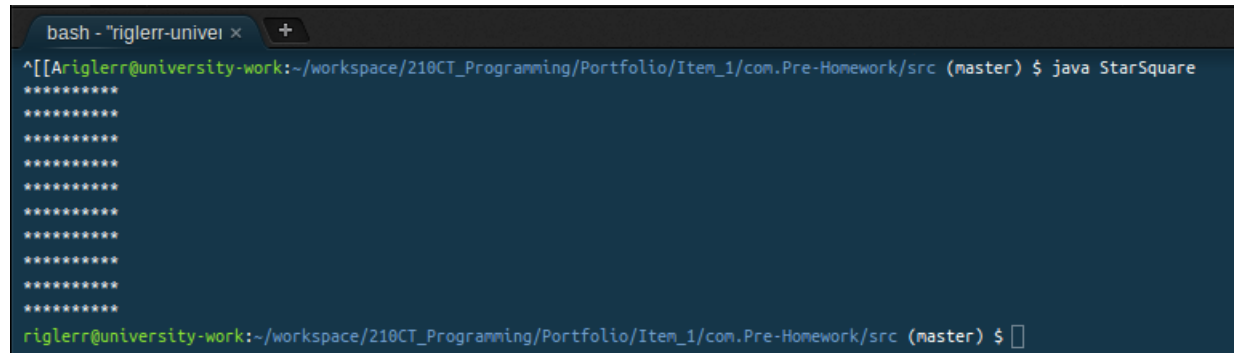
```
bash - "riglerr-univer x"
riglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $ java NameRepeat
1 Rob
2 Rob
3 Rob
4 Rob
5 Rob
6 Rob
7 Rob
8 Rob
9 Rob
10 Rob
riglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $
[riglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master)]
```

## Pre-Homework 2: Write a function that draws a square of stars given as a parameter

Listing 2: StarSquare class JAVA code

```
1  /**
2   * Created by Rob on 23/10/2014.
3   */
4  public class StarSquare {
5
6      //Create variable to hold asterisk character.
7      public char ast = '*';
8      public static void main(String[] args){
9
10         StarSquare sSquare = new StarSquare(); /*Create Object*/
11
12         /*Use object to call writeSquare() method*/
13         sSquare.writeSquare(10);
14
15     }
16
17
18     public void writeSquare(int size){
19
20         for(int i =0; i<size;i++){ /*OuterLoop 'size' times*/
21             for(int j = 0;j<size;j++){ /*InnerLoop 'size' times*/
22
23                 System.out.print(ast); /*Print line of asterisks*/
24
25             }
26
27             /* Start new line when inner loop finishes*/
28             System.out.println();
29         }
30     }
31 }
```

### Evidence



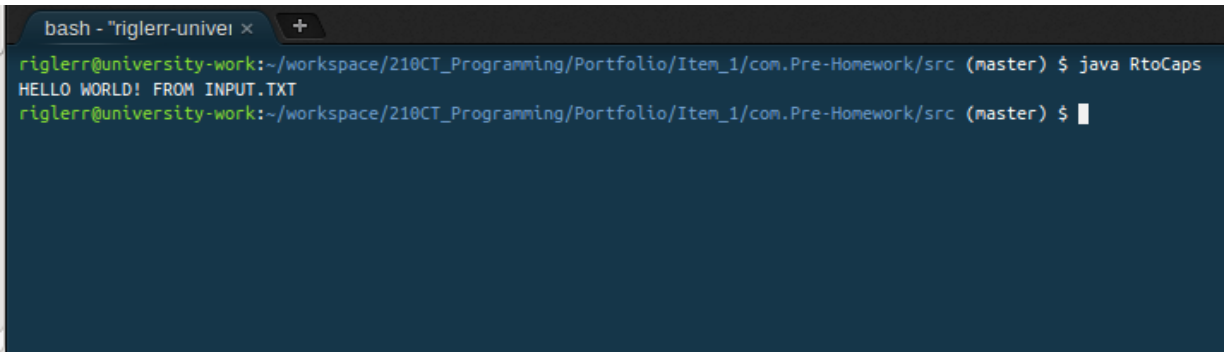
```
bash - "riglerr-univer × +
^[[Ariglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $ java StarSquare
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
riglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $
```

## Pre-Homework 3: Write a program to open a file and display it's contents in capitals

Listing 3: RtoCaps class JAVA code

```
1  /**
2   * Created by Rob on 23/10/2014.
3   */
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.util.Scanner;
7
8  public class RtoCaps {
9
10     public static void main(String[] args) throws FileNotFoundException
11     {
12         File inFile = new File("input.txt");
13         /*Create a file object */
14         RtoCaps obj = new RtoCaps(); /*Create class object*/
15
16         /*Use Class object to call rInput() method*/
17         obj.rInput(inFile);
18     }
19
20     public void rInput(File inFile) throws FileNotFoundException{
21
22         /*Create a new scanner to read from the file*/
23         Scanner in = new Scanner(inFile);
24
25         /*Loop WHILE there is still lines left in the document*/
26         while(in.hasNextLine())
27         {
28             /* Place the next line in a strin varibale*/
29             String line = in.nextLine();
30
31             /* Print the line in uppercase*/
32             System.out.println(line.toUpperCase());
33         }
34
35     }
36
37
38 }
```

### Evidence

A terminal window with a dark background. The title bar shows 'bash - "riglerr-univer' and a '+' icon. The prompt is 'riglerr@university-work:~/workspace/210CT\_Programming/Portfolio/Item\_1/com.Pre-Homework/src (master) \$'. The command 'java RtoCaps' has been executed, and the output is 'HELLO WORLD! FROM INPUT.TXT'. The prompt is now 'riglerr@university-work:~/workspace/210CT\_Programming/Portfolio/Item\_1/com.Pre-Homework/src (master) \$' with a cursor.

## 1. Pseudocode for linear search

---

### Algorithm 1 LinearSearch

---

```
1: procedure BOOL LINEARSEARCH(item, list[])
2:   for each element i in list do
3:     if list[i] = item then
4:       return true
5:     end if
6:   end for
7: return false
8: end procedure
```

---

## 2. Pseudocode for finding duplicates in a list

---

### Algorithm 2 Examining for duplicates

---

```
1: procedure BOOL EXFORDUPES(list[])
2:   for each element i in list[] do
3:     for each element j in list[] do
4:       if list[i] = list[j] then
5:         return true
6:       end if
7:     end for
8:   end for
9: end procedure
```

---



## Item 2: Week 4 - Time complexities and Big-O notation

### 1. Describe the runtime bounds of the linear search algorithm

---

**Algorithm 3** LinearSearch

---

```
1: procedure BOOL LINEARSEARCH(item, list[ ])
2:
3:   for each element i in list do           (n)
4:     if list[i] = list then t           (n)
5:       return true                         (n)
6:     end if
7:   end for
8: return false                             (1)
9: end procedure
```

---

The time complexity of the algorithm is  $O(n)$

### 2. Describe the runtime bounds of the duplicate finder algorithm

---

**Algorithm 4** Examining for duplicates

---

```
1: procedure BOOL EXFORDUPES(list[ ])
2:   for each element i in list[ ] do           (n)
3:     for each element j in list[ ] do           (n*n)
4:       if list[i] = list[j] then           (n*n)
5:         return true                         (n*n)
6:       end if
7:     end for
8:   end for
9: return false                             (1)
10: end procedure
```

---

The time complexity of the algorithm is  $O(n^2)$

**Additional work: Critical values of relative runtimes**

Write a function that determines the critical value at which the relative runtime of two linear algorithms swap.

---

**Algorithm 5** Relative runtime comparison algorithm

---

```

1: procedure CRITVAL( $m1, k1, m2, k2$ )
2:    $switch \leftarrow false$ 
3:    $n \leftarrow 0$ 
4:
5:   // Which Expression has a greater value for  $n=0$ 
6:   if  $((m1 * n) + k1) > ((m2 * n) + k2)$  then
7:
8:     //While Expression 1 ( $m1, k1$ ) is greater than Expression 2( $m2, k2$ ), do:
9:     while  $!switch$  do
10:
11:       // If Expression 1 become less than Expression 2 for that value of  $n$ 
12:       if  $((m1 * n) + k1) < ((m2 * n) + k2)$  then
13:
14:         //switch becomes true, which exits the loop and both if statements
15:          $switch \leftarrow true$ 
16:       else
17:          $n++$ 
18:       end if
19:     end while
20:
21:   else
22:
23:     //While Expression 2 ( $m1, k1$ ) is greater than Expression 1( $m2, k2$ ), do:
24:     while  $!switch$  do
25:
26:       // If Expression 2 become less than Expression 1 for that value of  $n$ 
27:       if  $((m1 * n) + k1) > ((m2 * n) + k2)$  then
28:
29:         //switch becomes true, which exits the loop and both if statements
30:          $switch \leftarrow true$ 
31:       else
32:          $n++$ 
33:       end if
34:     end while
35:
36:   end if
37:   // Return the value of  $n$  at which either while loop was fulfilled.
38:   return  $n$ 
39:
40: end procedure

```

---

## Item 3: Week 6 - Harmonic Series or Pivot Selection

### Pre-Homework 1.: 3-Bit binary number

Write a function that takes 3 boolean parameters,  $a, b$  and  $c$  and returns an integer value they represent if they are the three bits of a three-bit number, with  $a$  being the most significant and  $c$  being the least.

---

#### Algorithm 6 3-Bit Binary Number

---

```

1: procedure PRE1(bool  $a$ , bool  $b$ , bool  $c$ )
2:    $total \leftarrow 0$ 
3:   if  $a$  then
4:      $total \leftarrow total + 1$ 
5:   end if
6:   if  $b$  then
7:      $total \leftarrow total + 2$ 
8:   end if
9:   if  $c$  then
10:     $total \leftarrow total + 4$ 
11:  end if
12: return  $total$ 
13: end procedure

```

---

### Pre-Homework 2.: Boolean sequence binary number

Write a function that takes a sequence of values of any given length and returns the integer value they represent.

---

#### Algorithm 7 Return integer value from list a of boolean value

---

```

1: procedure PRE2(bool  $list[]$ )
2:    $total \leftarrow 0$ 
3:    $len \leftarrow \text{LengthOf}.list[]$ 
4:   for  $i \leftarrow len$  to 0 do
5:     if  $list[i] = \text{true}$  then
6:        $total \leftarrow total + 2^i$ 
7:     end if
8:   end for
9: end procedure

```

---

## 1. Harmonic Series (Pseudocode)

Use pseudocode to specify a recursive algorithm to compute the  $n$ th value of the harmonic series, for some integer  $n$ .

---

**Algorithm 8** Computing  $n$ th value of harmonic series

---

**procedure** HARM(float  $t$ , float  $n$ )    **while**  $n > 0$  **do**         $t \leftarrow t + (1/n)$          $n \leftarrow n - 1$         HARM( $t, n$ )    **end while****return**  $t$ **end procedure**

---

## 2. Harmonic Series (JAVA Implementation)

The Harmonic Series computation algorithm implemented in Java

Listing 4: Somethinbg

```
1 public class harms{
2
3     public static void main(String[] args){
4         /**/
5         System.out.println(f(0,3));
6     }
7
8     public static float f(float t, float n){
9         /*
10         t always has a value of 0 on the initial method call.
11         n is the nth term, which decreases by 1 each recursive call.
12
13         When n = 0, stop recursive calling and return the value t.
14         */
15         while (n>0)
16         {
17             t+= (1/n);
18             f(t,--n);
19         }
20         return t;
21     }
22
23
24 }
```

## **Item 4: Week 7 - Heapworksheet or RPN Calculator**

## **Item 5: Week 8 - Linked List Delete function or Linked List Sortings**