WEEK 7

ASSEMBLY OF TEARS

TODAYS PLAN

- Tutor Task
  - Factorial
  - Toolbox
  - Quicksort
  - Assembly

TUTOR TASKS

FACTORIAL

FACTORIAL – HEAD VS TAIL RECURSION

```
public void tail(int n)
{
   if(n == 1)
      return;
   else
      System.out.println(n);

   tail(n-1);
}
```

```
public void head(int n)
{
  if(n == 0)
     return;
  else
     head(n-1);

   System.out.println(n);
```

FACTORIAL – HEAD VS TAIL RECURSION

- Tail Recursion: Recursion call is at the end of the method
- Head Recursion: Recursion call is at the start of the method

FACTORIAL – HEAD VS TAIL RECURSION

- Tail Recursion: Recursion call is at the end of the method
- ~~Head Recursion: Recursion call is at the start of the method~~

FACTORIAL – HEAD VS TAIL RECURSION

- How to Remember:

  Tail Recursion: Nothing comes after the tail

FACTORIAL – HEAD VS TAIL RECURSION

- More Reading: https://www.cs.cmu.edu/~adamchik/15-121/lectures/Recursions/recursions.html

FACTORIAL

$$n! = \prod_{i=1}^{n} i \qquad (n \geq 1)$$

$$0! = 1$$

- Implement factorial using
  - Iterative
  - Head Recursion
  - Tail Recursion
    - Advisable to use a help method:
      - private static int facTailRecHelper(int n, int k)

TOOLBOX

TOOLBOX

- No loops allowed
  - No while
  - No for
- Use Recursion

TOOLBOX – ISEVEN

- Boolean isEven(int n)
  - True if Even
  - False if Odd

TOOLBOX – EVEN SUM

Public static int evenSum(int n)

- Sum all even numbers from n to 0 (also negatives)
  - F(8):= 8+6+4+2(+0) = 20
  - F(-8):= -8+-6+-4+-2(+-0) = -20
- No *, /, % allowed
- Do use isEven

TOOLBOX – MULTIPLICATION

Public static int multiplication(int x, int y)

- Implement multiplication via addition recursively

TOOLBOX – ARRAY REVERSION

Public static void reverse(int[] m)

- Reverse an array recursively
- Without the creation of temporary arrays

TOOLBOX – ARRAY REVERSION

Public static int numberofOddIntegers(int[] m)

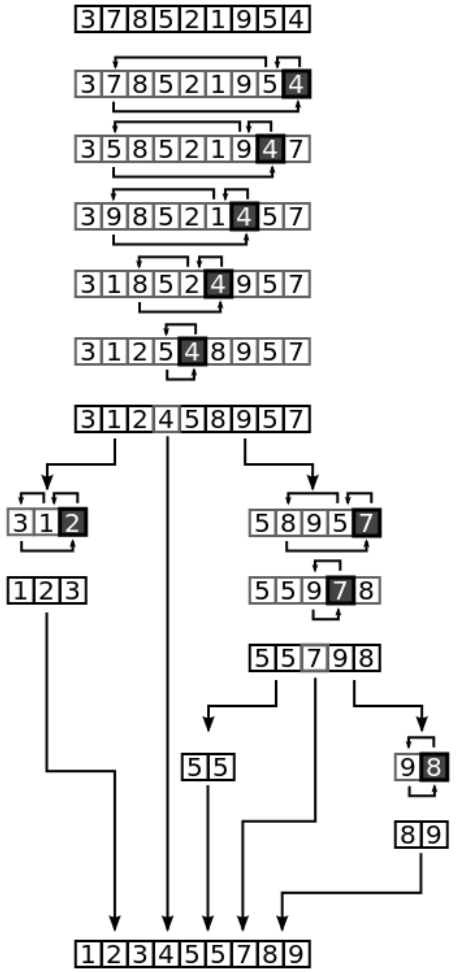
- Count recursively how many integers are odd

TOOLBOX – ARRAY REVERSION

Public static int filterOdd(int[] m)

- Create a new array with only the odd numbers
- In order of the original array

QUICKSORT


3 7 8 5 2 1 9 5 4
3 7 8 5 2 1 9 5 4
3 5 8 5 2 1 9 4 7
3 9 8 5 2 1 4 5 7
3 1 8 5 2 4 9 5 7
3 1 2 5 4 8 9 5 7
3 1 2 4 5 8 9 5 7
3 1 2
5 8 9 5 7
1 2 3
5 5 9 7 8
5 5 7 9 8
5 5
9 8
8 9
1 2 3 4 5 5 7 8 9

QUICKSORT – DEMO

- Doing Quicksort with [5 1 3 9 1 5 3]

QUICKSORT – IMPLEMENTATION

- Void swap(int[] numbers, int i, int j)
- Int partition (int[] numbers, int left, int right)
  - Numbers[right] = pivot element
  - Array between left and right inclusive of pivot should be swapped according to quicksort
- Implements Quicksort
- Write a suitable main method that generates a random array
  - Math.random()
  - ran Random = new Random();
    - From java.util.Random

ASSEMBLY

ASSEMBLY

- 10 Minutes Reading time for the homework
- Most details on assembly are written there

ASSEMBLY – LINE BY LINE

```
1  IN
2  IN
3  LDI ggt
4  CALL 2
5  OUT
6  HALT
```

- In gets input via a method like Read() and is placed in the stack
- A 16 Bit Address of ggt is loaded onto the stack
- Call the function with the two parameters (the order of which is the stack order)
- Output value at top of stack
- End Program

ASSEMBLY – LINE BY LINE

```
8  // 2 Argumente (a bei -1, b bei 0)
9  ggt:
10 ALLOC 1
```

- Function declaration
- Reserves space for 1 variable (1x 16 bits)

ASSEMBLY – LINE BY LINE

```
11  // Tausch von größerer Zahl nach vorne
12  LDS -1
13  LDS 0
14  JLT loop
15  LDS 0
16  LDS -1
17  STS 0
18  STS -1
```

- Copy variable 1 into the stack
- Copy variable 2 into the stack
- If var1<var2 Jump to Next Step
- Else
- Copy Var2 into the stack
- Copy Var1 into the stack
- Save Var2 to Var1's location
- Save Var1 to Var2's location

ASSEMBLY – LINE BY LINE

```
19 // Hauptschleife
20 loop:
21 LDS 0
22 STS 1
23 LDS 0
24 LDS -1
25 MOD
26 STS 0
27 LDS 1
28 STS -1
29 LDS 0
30 LDI 0
31 JNE loop
32 LDS -1
33 // Wir geben zwei Argumente und eine lokale Variable frei
34 RETURN 3
```

- Loop: Begins here
- Setup for modulo operation
- B = b%a
- Setup for Jump Not Equal
- If Not Equal, jump back to line 20
- Copy the result into the stack