Sửa Lab03

* Exercise **2c**

public static int countDigits(int n) {

if (n < 10) {

return 1;

}

return 1 + countDigits(n / 10);

}

* Exercise **3**

public static boolean **checkPrime**(int n, int d) {

if (n < 2)

return false;

if (n == 2) {

return true;

}

if (n % d == 0) {

return false;

}

if (n - 1 == d) {

return true;

}

return **checkPrime**(n, d + 1);

}

* Exercise **4a, 4d**

public static int **ex4\_a** (int n) {

if (n == 1)

return 3;

return (2 \* n + 1) + **ex4\_a**(n - 1);

}

public static int **ex4\_d** (int n, int r) {

if (n >= r&& r > 0) {

return (n - r + 1) \* **ex4\_d**(n, r - 1);

}

else

return 1;

}

* Exercise **5**

public static int **dec2bin**(int n) {

if (n == 0)

return 0;

return n % 2 + **dec2bin**(n / 2) \* 10;

}

* Exercise **7**

public static int **minElement**(int[] a, int k) {

if (k == 0)

return a[0];

return a[k] < **minElement**(a, k - 1) ? a[k] : **minElement**(a, k - 1);

}

public static int **sum**(int[] a, int k) {

if (k == 0) {

return a[0];

}

return a[k] + **sum**(a, k - 1);

}

public static int **countEven**(int[] a, int k) {

if (k < 0) {

return 0;

}

if (a[k] % 2 == 0) {

return 1 + **countEven**(a, k - 1);

}

return **countEven**(a, k - 1);

}

public static void main(String[] args) {

int[] a = {8, 2, 0, 1}

System.out.println(minElement(a, a.length - 1); // k = a.length - 1 = 3

// return a[2]

// ↑ return (pop)

// a[3] < minElement(a, 2) = a[2]

// gọi đệ quy (push) ⇵ return (pop)

// a[2] < minElement(a, 1) = a[1]

// gọi đệ quy (push) ⇵ return (pop)

// a[1] < minElement(a, 0)

// gọi đệ quy (push) ⇵ return (pop)

// a[0]

}

* Exercise **8**

public void **addSortedLinkedList**(Integer key) {

head = **addSortedLinkedList**(head, key);

}

private Node<Integer> **addSortedLinkedList**(Node<Integer> temp, Integer key) {

}

public int **countEven**(Node<Integer> tmp) {

if (tmp == null) {

return 0;

}

if (tmp.**getData**() % 2 == 0){

return 1 + **countEven**(tmp.**getNext**());

}

return **countEven**(tmp.**getNext**());

}

Nội dung thi GK: Lab 1 → Lab5

**Selection Sort**

min\_idx = i, j = i + 1

i = 0 8 2 0 1 9

→

0|2 8 1 9

0 1|8 2 9

0 1 2|8 9

0 1 2 8|9 → không cần xét phần tử cuối

**Bubble Sort**

8 2 0 1 9

2 0 1 8|9 → n - 1 - i

0 1 2|8 9

0 1|2 8 9

không cần xét phần tử cuối ← 0|1 2 8 9

**Insertion Sort**

không xét đầu

**↗**

**i = 1** 8| 2 0 1 9

key = 2, j = i - 1 = 0

8 8 0 1 9 (a[j + 1] = a[j])

2 8 0 1 9 (a[j] + 1 = key)

**i = 2** 2 8| 0 1 9

**i = 3** 0 2 8| 1 9

0 1 2 8| 9

0 1 2 8 9

Lưu ý: **shift** chứ không phải swap

Độ phức tạp: **O(n^2)**

Merge Sort

Quick Sort

Độ phức tạp: O(n logn)