Nalongsone Danddank Student ID : 14958950 StarID: jf3893pd

Email: [nalongsone.danddank@my.metrostate.edu](mailto:nalongsone.danddank@my.metrostate.edu)\

**Metropolitan State University**

**ICS-365-01 —Organization of Programming Languages**

**Homework #10**

1. Rewrite the following pseudocode segment using a loop structure in the specified languages:

k = (j + 13) / 27

loop:

**if** k > 10 **then goto** out

k = k + 1

i = 3 \* k - 1

**goto** loop

out: . . .

1. C or Java

for ( k = (j + 13)/27; k <= 10; i = 3 \* ++k - 1 );

b. Python

for k in range((j + 13)//27 + 1, 11):

i = 3 \* k -1

1. Rewrite the following code segment using a multiple-

Selection statement in the following languages:

**if** ((k == 1) || (k == 2)) j = 2 \* k - 1

**if** ((k == 3) || (k == 5)) j = 3 \* k + 1

**if** (k == 4) j = 4 \* k - 1

**if** ((k == 6) || (k == 7) || (k == 8)) j = k – 2

1. C or Java

switch ( k ) {

case 1:

case 2: j = 2 \* k - 1; break;

case 3:

case 5: j = 3 \* k + 1; break;

case 4: j = 4 \* k - 1; break;

case 6:

case 7:

case 8: j = k - 2; break;

default:

}

b. Python

if k == 1 or k == 2 :

j = 2 \* k -1

elif k == 3 or k == 5 :

j = 3 \* k + 1

elif k == 4 :

j = 4 \* k - 1

elif k == 6 or k == 7 or k == 8 :

J = k - 2

Assume all variables are integer type. **Discuss** (write) the relative merits of the use of these languages for this particular code.

1. Consider the following C program segment. Rewrite it using no gotos or **break**s.

j = -3;

**for** (i = 0; i < 3; i++) {

**switch** (j + 2) {

**case** 3:

**case** 2: j--; **break**;

**case** 0: j += 2; **break**;

**default**: j = 0;

}

**if** (j > 0) **break**;

j = 3 - i

}

Answer:

j = -3;

**for** (i = 0; i < 3; i++) {

**if**((j+2 == 3) ||(j+2 == 2))

j--;

**else** {

**if**(j + 2 == 0)

j += 2;

**else**

j = 0;

}

**if** (j <= 0)

j = 3 - i;

**else**

i = 3;

}

1. Consider the following programming problem: The values of three

Integer variables—first, second, and third—must be placed in the

three variables max, mid, and min, with the obvious meanings, without using arrays or user-defined or predefined subprograms. Write two solutions to this problem, one that uses nested selections and one using not-nested selection statements. **Compare (write) the complexity and expected reliability of the two.**

Solution:

nested selections: (Java)

**if** (first > second) {

**if** (first > third) {

max = first;

**if** (second > third) {

mid = second;

min = third;

} **else** {

mid = third;

min = second;

}

} **else** {

max = third;

mid = first;

min = second;

}

} **else** {

**if** (second > third) {

max = second;

**if** (first > third) {

mid = first;

min = third;

} **else** {

mid = third;

min = first;

}

} **else** {

max = third;

mid = second;

min = first;

}

}

not-nested selections:(Java)

**if** (first > second && first > third && second > third) {

max = first;

mid = second;

min = third;

}

**if** (first > second && first > third && third > second) {

max = first;

mid = third;

min = second;

}

**if** (second > first && second > third && first > third) {

max = second;

mid = first;

min = third;

}

**if** (second > first && second > third && third > first) {

max = second;

mid = third;

min = first;

}

**if** (third > first && third > second && first > second) {

max = third;

mid = first;

min = second;

}

**if** (third > first && third > second && second > first) {

max = third;

mid = second;

min = first;

}

Compare complexity and reliability of the two: I think **not-nested** selection is **less** complexity and more **reliable** than **nested** selection statements which write on java syntax above. Because **nested** selection statements seem like more complicated and less readable if compare to **not-nested** selection statements.