CSC B36 Additional Notes

pseudocode used for program correctness

⋆ Introduction

The pseudocode used in the course notes by Vassos Hadzilacos is based on the programming language Pascal. However, most B36 students today are most familiar with Python, having learned it in first year. For this course we will use Python-based pseudocode. So we present here the major differences between Pascal-based pseudocode and Python-based pseudocode. The reader should become familiar with both styles of pseudocode so as to be able to read and understand both course notes and lecture/tutorial material.

⋆ Assignments

Pascal uses := (colon equal) as its assignment operator. Python uses =.

E.g., a Pascal-based assignment statement like

```
$x := x + 2$
```

is the same as the Python-based assignment statement

```
$x = x + 2$
```

⋆ Loops

For the program correctness part of the course, we mainly concerned with the **while** loop. Pascal uses keywords **while**, **do** and **end while**. Python uses **while** and : (colon).

E.g., a Pascal-based loop like

```
while $x < n$ do
    $y := y * x$
    $x := x + 1$
end while
```

is the same as the Python-based loop

```
while $x < n$:
    $y = y * x$
    $x = x + 1$
```

⋆ If-then-else

For flow control, Pascal uses keywords **if**, **then**, **else** and **end if**. Python uses **if**, **elif**, **else** and : (colon).

E.g., some Pascal-based if-then-else code like

```
if $x = 0$ then
    $y := 0$
else if $x > 1$ and $x \bmod 2 = 0$ then
    $y := 2$
else
    $y := x * x$
end if
```

is the same as the Python-based if-then-else code

```
if $x == 0$:
    $y = 0$
elif $x > 1$ and $x \bmod 2 == 0$:
    $y = 2$
else:
    $y = x * x$
```

⋆ Arrays/Lists

A Pascal array is more or less the same as a Python list. A Pascal array A starts at index 1 and goes to index length(A). A Python list L starts at index 0 and goes to index len(L) $- 1$. Python also allows for a negative index as in $L[-j]$, where $1 \leq j \leq \text{len}(L)$, which refers to the j-th element from the end of list L. Pascal does not allow this.

To get a subarray, the course notes (based on Pascal) uses $A[i..j]$ to mean the part of the array A from index i to index j, *including both* $A[i]$ *and* $A[j]$. In class we will use the Python slice notation, where $L[i:j]$ means the part of the list L from index i to index $j-1$, *including* $L[i]$ *but not* $L[j]$.

E.g., let P be an array/list containing numbers $[2, 3, 5, 7, 11, 13, 17, 19]$.
Then in Pascal-based notation,

$P[3..6]$ is $[5, 7, 11, 13]$ and $P[3..2]$ is the empty subarray,

and in Python-based notation,

$P[2:6]$ is $[5, 7, 11, 13]$ and $P[2:2]$ is the empty sublist.

Pascal allow two-dimensional arrays (e.g., to represent a matrix). Python essentially treats these as lists of lists. Pascal uses $M[i, j]$ to mean the (i, j)-th element of "matrix" M, while Python uses $M[i][j]$.

⋆ Strings

Pascal strings can thought of as arrays of characters. Similarly we will think of Python strings as lists of characters.

We will disregard the fact that Python strings are immutable. Instead we will always treat strings as mutable.

E.g., both the Pascal-based

$S[i]$:= 'B'

and the Python-based

$S[j]$ = 'C'

are allowed.

⋆ Multiple statements in a line

Independent of using Pascal or Python, we will use ; (semicolon) as a delimiter to separate statements within a single line.

E.g., the Pascal-based statements

```
$y := y * x$
$x := x + 1$
```

can be put into a single line as

```
$y := y * x$;     $x := x + 1$
```

Similarly, the Python-based statements

```
$y = y * x$
$x = x + 1$
```

can be put into a single line as

```
$y = y * x$;     $x = x + 1$
```