Computer Science COMP-102B
OPEN • BOOKS • / • OPEN • NOTES

1) a) How many CDs can we store on a double layer Blu-Ray disc?

b) What is the binary representation of the integer 377?

c) What number has (32 bits) floating point representation 10010101010101010010101101010110?

d) In the internet section we saw two notations to extract the network address from an arbitrary IP address. We saw an example with both notations: “/19” and network mask “255.255.224.0”. Explain how these two notations are equivalent and what they mean.

e) My computer has an Ethernet Address of 00:1B:63:C4:08:4E Please write the binary equivalent of this address.

2) a) Remember the algorithms we saw in class for finding minimum and sorting:

```
Procedure FindMin(x_1, x_2, ..., x_n)
mini:=1; min:=x_i
for i:=2 to n do
  if x_i<min then min:=x_i; mini:=i
output mini
```

```
input x_1 x_2 x_3 ... x_n
for i:=1 to n-1 do
  j:=i-1+FindMin(x_i x_{i+1} ... x_n)
  temp:=x_i; x_i:=x_j; x_j:=temp
output x_1 x_2 x_3 ... x_n
```

Simulate the execution of the sorting algorithm on input (x_1=5 x_2=2 x_3=1), including all executions of FindMin.

b) In the above algorithm for sorting I wrote explicitly the instructions for swapping two variables x_i, x_j as follows:

```
temp:=x_i; x_i:=x_j; x_j:=temp
```

This way of swapping uses an extra variable temp to store the first value while we replace it with the second. The following instructions swap without an extra variable

```
x_i:=x_i+x_j; x_j:= x_i-x_j; x_i:= x_i-x_j
```

Explain why this works and find some disadvantage to swapping this way.
3) Remember the following algorithm from HW2:

```plaintext
Input T, x1, x2, ..., xn
Ti := 0
for i := 1 to n do
    if T = xi then Ti := i
output Ti
```

Rewrite this algorithm recursively so that it terminates as soon as an occurrence of T is found (return the first occurrence not the last as in the above algorithm), but of course still returns 0 if no occurrence is found.

4) Remember the algorithm we saw in class for adding two sequences of numbers base B:

```plaintext
input B, x_n, x_{n-1}, ..., x_0, y_n, y_{n-1}, ..., y_0
carry := 0
for i := 0 to n do
    Bigit := x_i + y_i + carry
    if Bigit ≥ B then z_i := Bigit - B; carry := 1
    else z_i := Bigit; carry := 0
z_{n+1} := carry
output z_{n+1}, z_n, z_{n-1}, ..., z_0
```

a) Why does it make sense to use operations like + and - in an algorithm supposed to implement addition?

b) Argue that if we assign Bigit := x_i + y_i + carry where x_i, y_i are less than B and that carry ≤ 1 then Bigit < 2B which in turns imply \(-B < (z_i := Bigit - B) < B\).

c) Simulate this algorithm with inputs B = 5, x_i = 4, x_0 = 3, y_i = 3, y_0 = 2.