CAS CS 320
Lec 12

Runtime & Continuation

1. Runtime
    a. fun rev(xs) =
        case xs of
        nil => nil
        | x1 :: xs => rev(xs) @ [x1]
    b. T(n) = T(n-1) + O(n)
       T(n) is O(n^2)
    c. fun append(xs,ys) =
        case xs of
        nil => ys
        | x1 :: xs => x1 :: append(xs,ys)
    d. T(n) = T(n-1) + O(1)
       T(n) = O(n)
    e. fun rappend(xs,ys) =
        case xs of
        nil => ys
        | x1::xs => rappend(xs, x1::ys)
    f. T(n) = T(n-1) + O(1)
       T(n) is O(n)
    g. fun ghaap(xs) =
        case xs of
        nil => nil
        | x1 :: xs => x1 :: ghaap(xs)
    h. T(n) = T(n-1) + O(1)
       T(n) is O(n)
    i. fun ghaap(xs) =
        case xs of
        nil => nil
        | x1 :: xs => x1 :: ghaap(ghaap(xs))
    j. T(n) = T(n-1) + T(n-1) + O(1)
       T(n) = 2T(n-1) + O(1)
       T(n) is O(2^n)
    k. fun ghaap(xs) =
        case xs of
        nil => nil

| x1 :: xs => ghaap(ghaap(xs))

l. T(n) = T(n-1) + O(1)
   T(n) is O(n)
m. Mergesort → split the list into two lists of same size, the splitting takes O(n), merging the two lists again takes O(n)
   T(n) = T(n/2) + T(n/2) + O(n) + O(n)
   T(n) = 2*T(n/2) + 2O(n) = 2*T(n/2) + O(n)
   T(n) = O(n * log(n))
n. Quicksort
   i. Worst case
      n n-1 1
         n-k k
      T(n) = T(n-1) + O(n)
      T(n) is O(n^2)
o. fun alter(xs, ys) =
      case xs of
      nil => nil
      | x1 :: xs => x1 :: alter(ys,xs)
p. T(0, n) = O(1)
   T(m, n) = T(n, m-1) + O(1)
   T(m, n) = O(m + n)
q. O(max(m,n)) is same as O(m+n)
r. Matrix Multiplication
   i. A is n * n
   ii. B is n * n
   iii. A * B is n * n
   iv. O(n^3)
s. datatype btree = btree_nil | btree_cons of (‘a btree) * ‘a * (‘a btree)
   i. 2^(min height) < size < 2^(max height)
t. fun flatten(xs) =
      btree_nil => nil
      btree_cons(t1, x0, tr) => flatten(t1) @ [x0] @ flatten(tr)
u. T(n) = 2T(n/2) + O(n)
   T(n) = O(n*log(n))

2. Continuation
   a. fun fact(x) =
         if x > 0 then x * fact(x-1) else 1
   b. fun kfact(x,k) =
         if x > 0 then kfact(x-1, fn res => kfact(x * res))
         else k(1)

```
val fact10 = kfact(10, fn res => res)
val _ = kfact(10, fn res => print("res = " ^ Int.toString(res) ^ "\n"))
```

c.
```
fun fibo(x) =
    if x < 2 then x
    else fibo(x-2) + fibo(x-1)
```

d.
```
fun kfibo(x,k) =
    if x < 2 then k(x)
    else kfibo(x-2, fn res1 => kfibo(x-1, fn res2 => k(res1 + res2)))
```

e.
```
fun f91(x) =
    if x > 100 then x - 10
    else f91(f91(x+11)
```

f.
```
fun kf91(x) =
    if x > 100 then k(x-10)
    else kf91(x+11, fn res => kf91(res,k))
```

g.