CAS CS 320
Lec 7

Higher Order Functions (cont.)

1. Higher order functions
    a.
    ```
    fun list_foreach(xs: ‘a list, work: ‘a -> unit): unit =
    (* build everything off of this function *)
            case xs of
            nil => ()
            | x1::xs => (work(x1),list_foreach(xs,work))
    ```
    b.
    ```
    fun list_forall(xs: ‘a list, test: ‘a -> bool): bool =
    (*if the test function always returns false, then the list_forall function checks
    whether the list is empty or not *)
            case xs of
            nil => true
            | x1::xs => if test(x1) = true then list_forall(xs, test) else false
    ```
2. SML library
    a.
    ```
    fun list_forall(xs: ‘a list, test: ‘a -> bool): bool =
            case xs of
            nil => true
            | x1 :: xs => test(x1) andalso list_forall(xs, test)
    ```
    b.
    ```
    fun list_exists(xs: ‘a list, test: ‘a -> bool): bool =
            case xs of
            nil => false
            | x1::xs => if test(x1) = false then list_exists(xs, test) else true
    OR
    fun list_exists(xs: ‘a list, test: ‘a -> bool): bool =
            case xs of
            nil => false
            | x1::xs => test(x1) orelse list_exists(xs, test)
    OR
    ```
    c.
    ```
    fun list_foreach(xs: ‘a list, work: ‘a -> unit): unit =
    let
            val _ = list_forall(xs, fn(x1) => (work(x1); true)) in ()
    end
    ```
    d.
    ```
    fun list_forall(xs: ‘a list, test: ‘a -> bool): bool =
    let
    exception False
    in
    ```

```
    list_foreach(xs, fn(x1) => if test(x1) then () else raise False); true)
    handle False => false
    end
e.  (* Third order function *)
    fun foreach_to_forall(foreach: (‘xs * (‘x0 -> unit)) -> unit):
    (‘xs * (‘x0 -> bool)) -> bool) =
    fn (xs: ‘xs, test: ‘x0 -> bool =>
    let
            exception False
    in
            let
                    val() = foreach(xs, fn(x0:’x0) =>
                                    if test(x0) then () else raise False)
            in
                    (true)
            end
            handle False => (false)
    end
f.  fun forall_to_foreach(forall: (‘xs * (‘x0 -> bool)) -> bool):
    (‘xs * (‘x0 -> unit)) -> unit =
    fn(xs, work) =>
    (forall(xs, fn(x0) => (work(x0); true));())

    fun list_foreach(xs, work) = forall_to_foreach(list_forall)(xs, work)
g.  fun foldleft_to_length(foldleft:(‘xs * int * (int * ‘x0 -> int)) -> int):
    (‘xs -> int) =
    fn (xs) => foldleft(xs, 0, fn(r0,x0) => r + 1)
h.  fun foreach_tolength(foreach) =
    foldleft_to_length(foreach_to_foldleft)
i.  fun string_foreach(xs: string, work: char -> unit): unit =
    int1_foreach(String.size(xs), fn(i) => work(String.sub(xs,i)))
j.  Then,
    foreach_tolength(string_foreach)(“abcde”) gives 5
k.  fun list_labelize(xs: ‘a list):
    (int * ‘a) list =
    list_reverse(#2(list_foldl(xs, (0, nil), fn((i,r), x) => (i+1, (i,x) :: r))))
```

l.
```
fun list_zip (xs: ‘a list, ys: ‘b list): (‘a * ‘b) list =
        case (xs, ys) of
        (nil, _) => nil
        | (_, nil) => nil
        | (x1::xs, y1::ys) => (x1,y1)::list_zip(xs,ys)
```
m.
```
fun list_z2foreach(xs: ‘a list, ys: ‘b list, work:(‘a * ‘b) -> unit): unit
        case (xs,ys) of
        (nil, _) => ()
        | (_, nil) => ()
        | (x1::xs, y1::ys) => (work(x1,y1);list_z2foreach(xs,ys,work))
```
3. Python library
    a.
    ```
    for (i,x) in enumerate(xs):
            work(i, x)
    ```
    b.
    ```
    xs = [x for x in range(10)]
    ys = [10-x for x in range(10)]
    xys = list(zip(xs,ys))
    (* zip pairs the corresponding elements in xs and ys *)
    ```