PPL – assignment 5

Part 1:

Question 1.1 a

(The code is also in ex5.rkt it was added here because it is relevant to part b)

Question 1.1 b

We will prove append$ is CPS-equivalent to append by induction on list1’s length   
(indicted by *n*). To do so we will prove that:

For every list1, list2 lists, and any continuation function (named *cont* for short)   
append$ is CPS-equivalent to append.

i.e.:

**Induction basis:** n=0, list1 is empty.

**Induction assumption:** let us assume, that for , s.t list1 length is n,   
the equation is valid:

**Induction step:** we will prove that the equation is valid for n+1:

From the assumption of induction for (with length n):

Part2:

Question 2 d

**The reduce1-lzl:** is good for computations which involve finite lazy-list.

**The reduce2-lzl:** is good for computations which involve infinite lazy-list, but we know exactly how many items to compute.

**The reduce3-lzl:** is good for approximations. (e.g., generate-pi-approximations in part f)

Question 2 g

**Advantage of implementation via *generate-pi-approximations* over *pi-sum*:**  
Every time we generate the next computation, without opening new frames (using tail recursion optimization), therefore we can avoid stack overflow.

**Advantage of implementation via *pi-sum* over *generate-pi-approximations*:**

We cat set the b member – which responsible for the precision of our approximation’s computation.

Part 3:

Question 1 a (

Question 1 b

**Diagram

Description automatically generated with medium confidence**Question 3