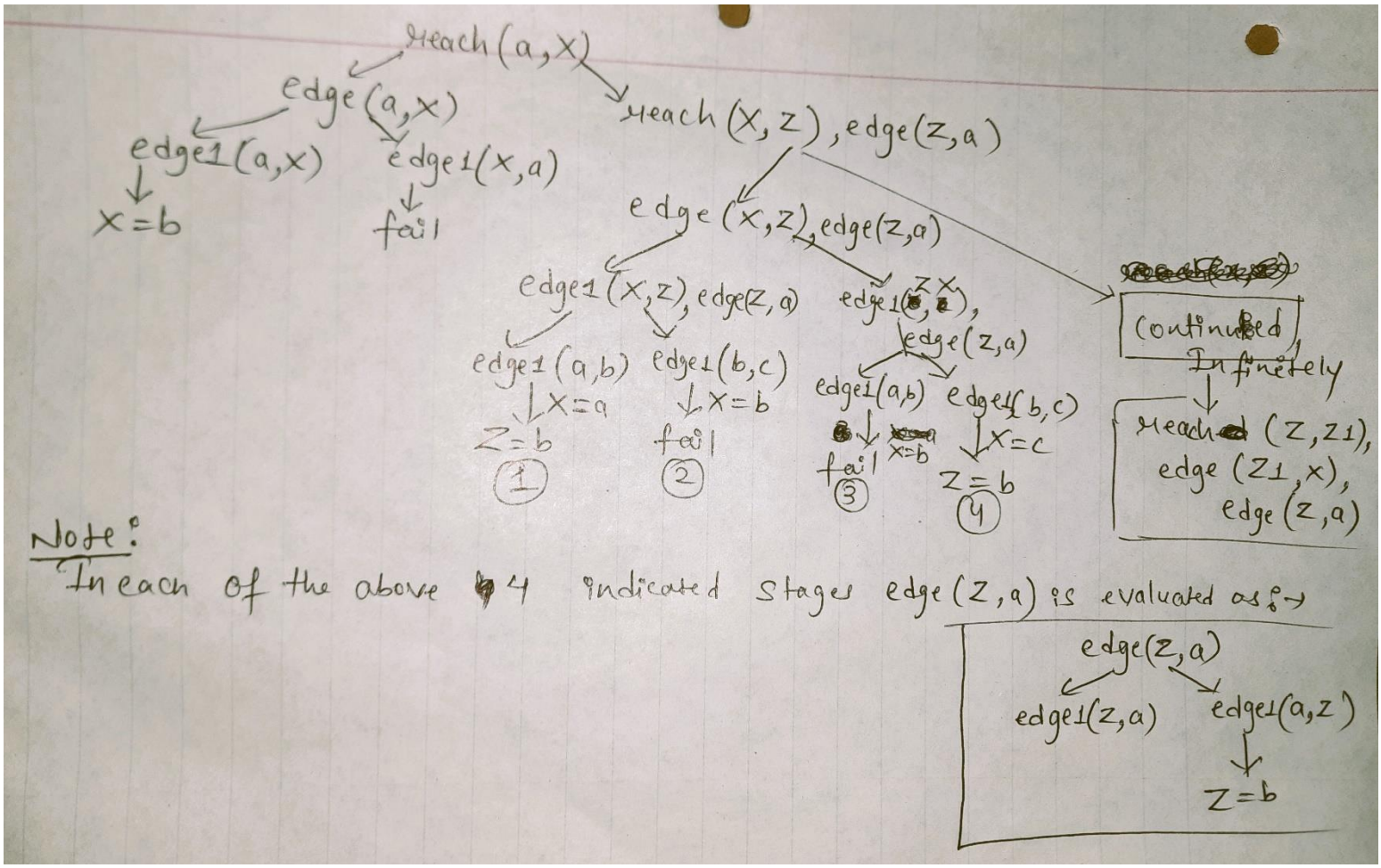
Assignment:4

Answer1:



Answer2:

$(\lambda x.\, x\ (\lambda m.(\lambda n.\lambda y.\, y))\ (\lambda u.\lambda v.\, u)\ (\lambda a.\lambda b.\, a\ b)$

→ Since $e_1\ e_2\ e_3 = (e_1\ e_2)\ e_3$ we can apply $(\lambda m.(\lambda n.\lambda y.\, y))$ as argument to $(\lambda x.\, x)$ which gives us:

$((\lambda m.(\lambda n.\lambda y.\, y))\ (\lambda u.\lambda v.\, u))\ (\lambda a.\lambda b.\, a\ b)$

→ Now applying $(\lambda u.\lambda v.\, u)$ as argument to $(\lambda m.(\lambda n.\lambda y.\, y))$, we get:

$(\lambda n.(\lambda y.\, y))\ ((\lambda a.(\lambda b.\, a))\ b)$

→ Now applying b as an argument to $(\lambda a.(\lambda b.\, a))$ gives:

$(\lambda n.(\lambda y.\, y))\ (\lambda b.\, b)$

→ Applying $(\lambda b.\, b)$ as argument to $(\lambda n.(\lambda y.\, y))$

$\lambda y.\, y$

Cannot be reduced further.

Answer 3(a):

- *** QoS_Compute**: It is a pointer to a function that computes the Quality of Service (QoS). The function computes QoS to enable Green to calculate the loss in QoS that arises from early loop termination. It is used to quantify the impact of approximations have on the QoS of the program.

  The programmer-supplied QoS_Compute() function is used in the calibration phase to tabulate the loss in QoS resulting from early loop termination at loop iteration counts specified by Calibrate_QoS. QoS Compute() function has the following interface: QoS Compute (return QoS, loop count, calibrate, Calibrate QoS, ...)

- **Calibrate_QoS**: This value is used in calibration phase and it specifies the loop iteration counts needed for calibration.

- **QoS_SLA**: This value specifies the QOS Service Level Agreements. It is used in conjuction with QoS model to make approximation decisions. The approximation uses the QoS SLA to determine appropriate intervals at which to measure change in QoS and the amount of QoS improvement needed to continue iterating the loop.

- **Sample_QoS**: It represents the sampling rate to perform runtime re-calibration. Since the QoS degradation on actual program inputs may differ from that observed during calibration, Green also samples the QoS loss observed at runtime and updates the approximation decision logic to meet the specified QoS target. If the programmer wants to avail of runtime re-calibration she must provide the sampling rate (sample_QoS) to perform re-calibration.

- **Static**: It indicates whether the programmer wants to use static or adaptive approximation. QoS Approximation QoS_Approx() comes in two main flavors for loop approximations, static and dynamic. In the static variety, the approximation is solely determined by the QoS model constructed in the calibration phase.

Answer 3(b):

The Green compiler first generates a “calibration” version of the program that is run with user-provided calibration inputs to generate QoS data needed to build the QoS model. To generate this version of the program calibration code is used.

Then the compiler uses this constructed QoS model to generate an “approximate” version of the program that can be run in place of the original. It synthesizes code to implement QoS Approx() and QoS ReCalibrate(). To generate the approximate version of the program, approximate code is used after the calibration code.

Answer 3(d):

The goal of QOS re-calibration mechanism is to ensure that the desired QoS is met. In contrast, a static approach to estimating QoS would have to account for non-linear effects arising from combining multiple approximations. The program’s behavior may occasionally differ from that observed on its training inputs and QoS ReCalibrate() provides a mechanism to detect and correct for this effect.

In the case of loop approximation, when used with static approximation re-calibration can update the QoS model and increase the loop iteration count threshold to compensate for higher than expected QoS degradation or decrease this threshold to improve performance and energy consumption when QoS degradation is lower than expected.

Similarly, when used with adaptive approximation re-calibration can appropriately change the interval used to measure change in QoS and/or QoS improvement required to continue. For function approximation, re-calibration allows Green to switch the approximate version of the function used to one that is more or less precise as determined by the observed QoS loss.

Answer 3(e):

Figure 14 demonstrates the effectiveness of Green’s re-calibration mechanism for Bing Search. It shows how can Green provide robust QoS guarantees even when supplied with an inaccurate QoS model. This inaccurate QoS model is

supplied for an experiment in which user indicates his/her desired QoS target as 2%, but the constructed QoS model incorrectly supplies M = 0.1N (which typically results in a 10% QoS loss).

After processing every 10K queries, Green monitors the next 100 consecutive queries (i.e., Sample QoS=1%) by running the precise version while also computing the QoS loss, if it had used the approximated version for those 100 queries. Since the current QoS model is not accurate enough, the monitored results will keep reporting low QoS. Then, Green's re-calibration mechanism keeps increasing the accuracy level (i.e., by increasing the M value by 0.1N) until it satisfies the user-defined QoS target. **The black line represents the QoS loss of the experiment with re-calibration which goes down as explained above and the grey line shows the number of documents processed by the approximate version which keeps on increasing as the value of M is increased by 0.1 everytime.**

Answer 3(f):

- Bing Search: In this case, QOS is defined as the percentage of queries that either return a different set of top N documents or return the same set of top N documents but in a different rank order, as compared to the base version

- 252.eon: For QOS in this case, we compute the average normalized difference of pixel values between the precise and approximate versions.

- Cluster GA: QOS defined as the normalized difference in the execution time of a parallel program scheduled by the base and approximate versions.

- Discrete Fourier Transform: QoS is defined as the normalized difference in each output sample of DFT between the precise and approximated versions.

- blackscholes: QoS is defined as the difference in option prices produced by the precise and approximate versions of the programs.