Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.

Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.

Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.

Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.

Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.

Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.

Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.

Helal Chowdhury HW3

Q1) 1) The worst case runtime of reverse1(lst) would be $\underline{\theta(n^2)}$. This is because the worst case for insert is $\theta(n)$ and the while loop runs for a length of n. so $\underbrace{n+n+n\ldots}_{n \text{ times}}$ is $\theta(n^2)$

2) The worst case running time of reverse2(lst) would be $\underline{\theta(n)}$. This is because the worst case for append is $\theta(1)$. This occurs a total of n times so $(1+1+1+1\ldots)\times n = \theta(n)$

Q3) The worst case running time of the implementation would be $\theta(n)$. This is because the first loop would run in $\theta(n)$ time, the second for loop runs also in just $\theta(n)$ time, because appending is usually constant. So therefore, the runtimes is $\theta(2n)$, or $\underline{\theta(n)}$.

Q4) a) The worst case scenario of run time would be $\theta(n^2)$. The worst case running time of remove is $\theta(n)$. If a list had a single element repeated n times and was removed, the while loop would run n times. Hence $n+n+n\ldots\}$ n times is $\theta(n^2)$

b) The worst case implementation of my code is $\theta(n)$. The for loop would merely run for $\theta(n)$, and the while loop only contains basic assigning functions so it is also $\theta(n)$. The final for loop runs in $\theta(n)$ too since pop is a constant time function. Therefore the implementation remains within $\theta(n)$.