1. Lakukan penelusuran terhadap eksekusi statement: head = insertEnd(head, 14);

   Diskusikan:

   o Urutan pemanggilan fungsi

   insert(head = head_awal, 14)
   → insert(head = head_awal->next, 14)
   → insert(head = head_awal->next->next, 14)
   → insert(head = head_awal->next->next->next, 14)
   → insert(head = head_awal->next->next->next->next, 14)
   → newNode(14)

   o Nilai parameter head pada setiap pemanggilan

   1. insert(head = head_awal, 14)
      Head merujuk ke node pertama yang megang data 6
   2. insert(head = head_awal->next, 14)
      Head merujuk ke node kedua (node setelah node pertama) yang megang data 8
   3. insert(head = head_awal->next->next, 14)
      Head merujuk ke node ketiga (node kedua setelah node pertama) yang megang data 10
   4. insert(head = head_awal->next->next->next, 14)
      Head merujuk ke node keempat (node ketiiga setelah node pertama) yang megang data 12
   5. insert(head = head_awal->next->next->next->next, 14)
      Head merujuk ke null pointer (setelah node keempat gak ada node lagi)

   o Kapan node baru dibuat

   Ketika head merujuk ke null pointer, dalam kata lain ketika head sudah melewati node terakhir

o Bagaimana proses unwinding terjadi hingga kembali ke main()

1. insert(head = head_awal->next->next->next->next, 14) selesai
   node(14)
   node(6) → node(8) → node(10) → node(12)

2. insert(head = head_awal->next->next->next, 14) selesai
   node(6) → node(8) → node(10) → node(12) → node(14)

3. insert(head = head_awal->next->next, 14) selesai
   node(6) → node(8) → node(10) → node(12) → node(14)

4. insert(head = head_awal->next, 14) selesai
   node(6) → node(8) → node(10) → node(12) → node(14)

5. insert(head = head_awal, 14) selesai
   node(6) → node(8) → node(10) → node(12) → node(14)

2. Berapa jumlah stack frame yang terbentuk saat menambahkan 14?
   a. insert(head = head_awal, 14)
   b. insert(head = head_awal->next, 14)
   c. insert(head = head_awal->next->next, 14)
   d. insert(head = head_awal->next->next->next, 14)
   e. insert(head = head_awal->next->next->next->next, 14)
   f. newNode(4)

   jawabannya 6

3. Apa kompleksitas waktu dan ruang dari insertEnd()? Jelaskan berdasarkan hasil tracing.
   Kompleksitas waktu:
   O(n) karena perlu geser satu per satu (traverse) semua N node yang ada dari depan ke belakang
   Kompleksitas ruang:
   O(n) karena perlu menyimpan 6 atau n+1 stack frame. O(n+1) is really just O(n).

4. Lakukan penelusuran terhadap eksekusi statement: traverse(head);

   Diskusikan:

   o Urutan pemanggilan fungsi

   traverse(head = head_awal)
   → traverse(head = head_awal->next)
   → traverse(head = head_awal->next->next)
   → traverse(head = head_awal->next->next->next)
   → traverse(head = head_awal->next->next->next->next)
   → traverse(head = head_awal->next->next->next->next->next)

   o Isi call stack

   traverse(head = head_awal->next->next->next->next->next)

   traverse(head = head_awal->next->next->next->next)

   traverse(head = head_awal->next->next->next)

   traverse(head = head_awal->next->next)

   traverse(head = head_awal->next)

   traverse(head = head_awal)

   o Urutan output yang dihasilkan

   6 8 10 12 14

5. Mengapa urutan output sesuai dengan urutan node pada linked list?
   Karena ngeprint dari depan, terus geser ke setelahnya, terus ngeprint yang ini, terus geser ke setelahnya lagi, gitu terus sampai setelahnya gak ada

6. Apakah traverse() juga memiliki kompleksitas ruang O(n)? Mengapa?
   Ya karena fungsi traverse() mengtraverse satu per satu node secara rekursi. Meskipun rekursi gak di bagian return, fungsi traverse(head) bakal tetep nungguin traverse(head->next) selesaai dulu.

7. Mengapa linked list cocok diimplementasikan secara rekursif?
   Karena linked list itu sifatnnya berpola. Sebuah problem dapat membuat subproblem yang sama persis kayak problem itu sendiri. Misal fungsi traverse(). Traverse() itu kan dia cetak data yang ada di node terus geser. Abis geser? Cetak data yang ada di node

terus geser. Abis geser? Cetak data yang ada di node terus geser. Karena polanya gitu2 aja, rekursi jadi cocok buat diimplementasiin.

8. Jika jumlah node sangat besar (misalnya 100.000), apa risiko penggunaan rekursi? Memorinya bakal penuh bos. Ingat rekursi itu bakal menuhin call stack dengan stack frame. Suatu fungsi rekursi (problem) baru bisa selesai kalau fungsi yang sama dipanggil oleh fugnsi rekursi itu (subproblem) udah selesai.