 Answers to review questions from Chapter 13

1. True or false: The computational complexity of a program depends only on its algorithmic structure, not on the structures used to represent the data.

**False. Both are important factors**

2. What does *wysiwyg* stand for?

**What you see is what you get.**

3. In your own words, describe the purpose of the buffer abstraction used in this chapter.

**The EditorBuffer class defines an expandable storage area for a text string that also specify a *cursor position* that indicates where the editing operations take place.**

4. What are the six commands implemented by the editor application? What are the corresponding public methods in the **EditorBuffer** class?

**F moveCursorForward**

**B moveCursorBackward**

**J moveCursorToStart**

**E moveCursorToEnd**

**I insertCharacter**

**D deleteCharacter**

5. In addition to the methods that correspond to the editor commands, what other public operations are exported by the **EditorBuffer** class?

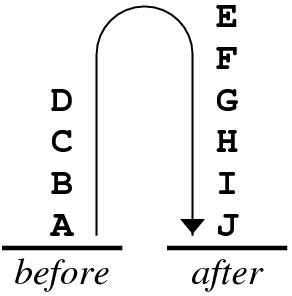
**The methods getText and getCursor.**

6. Which editor operations require linear time in the array representation of the editor buffer? What makes those operations slow?

**The methods insertCharacter and deleteCharacter. Both operations require moving characters in the array, either to make space for an inserted character or to close up the space resulting from a deletion.**

7. Draw a diagram showing the contents of the **before** and **after** stack in the two‑stack representation of a buffer that contains the following text, with the cursor positioned as shown:

/Users/eroberts/Books/CS2-in-C++/Chapters/13-EfficiencyAndRepresentation/pictures/EditorDiagrams/ABCDEFGHIJ4.png



8. How is the cursor position indicated in the two‑stack representation of the editor buffer?

**The cursor in the two‑stack editor is represented implicitly as the boundary between the two stacks.**

9. Which editor operations require linear time in the two‑stack representation?

**The methods moveCursorToStart and moveCursorToEnd.**

10. What is the purpose of the dummy cell in a linked list used to represent the editor buffer?

**The dummy cell eliminates the need for special‑case checking in the implementation because there is always a cell that can serve as the target of the cursor pointer.**

11. Does the dummy cell go at the beginning or the end of a linked list? Why?

**The dummy cell goes at the beginning. Inserting a new cell into the linked list requires maintaining a pointer to the cell before the insertion point, which means that insertion at the beginning of a buffer corresponds to inserting a new cell after the dummy cell.**

12. What are the five steps required to insert a new character into the linked‑list buffer?

**1. Allocate space for a new cell, and store the pointer to this cell in the temporary variable cp.**

**2. Copy the character to be inserted into the ch field of the new cell.**

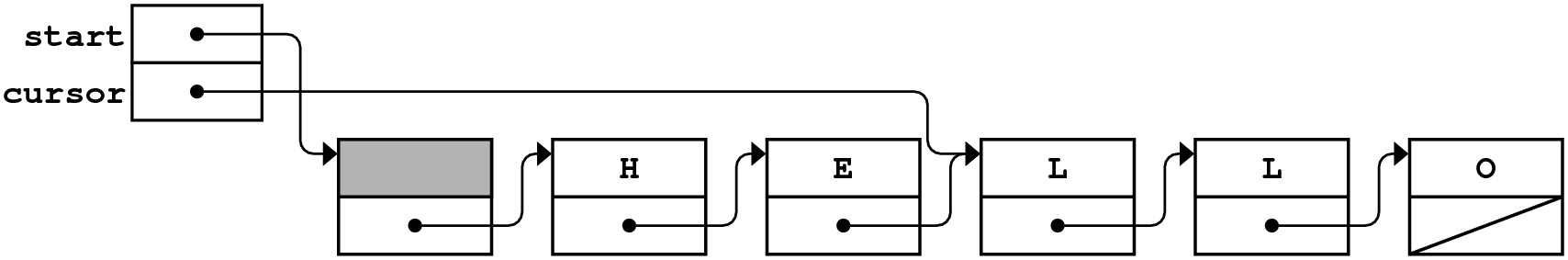
**3. Go to the cell indicated by the cursor field of the buffer and copy its link field to the link field of the new cell.**

**4. Change the link field in the cell addressed by the cursor so that it points to the new cell.**

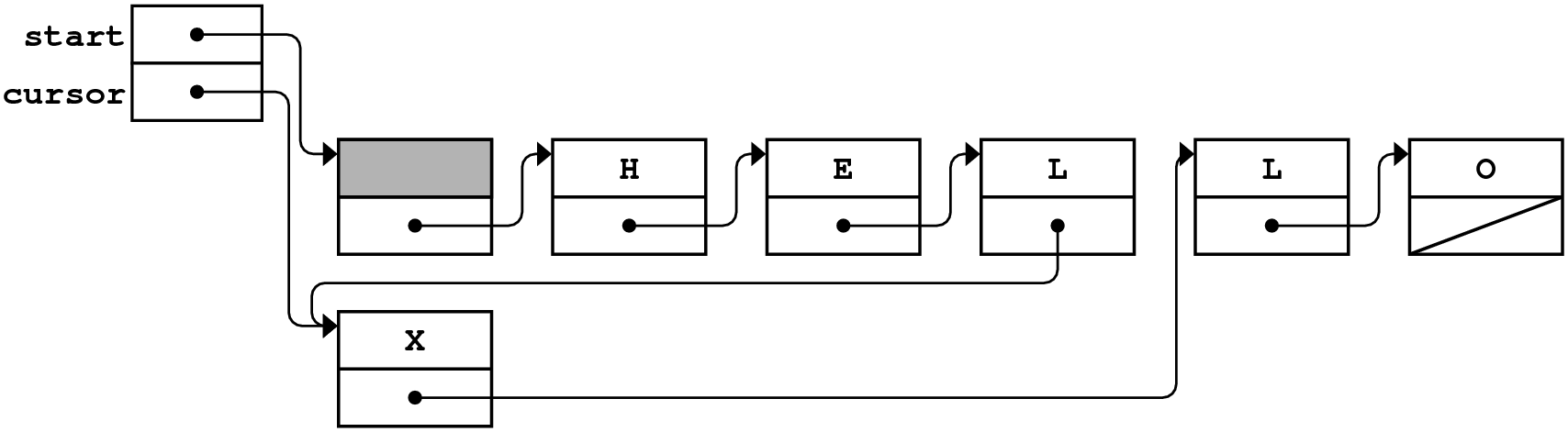
**5. Change the cursor field in the buffer so that it also points to the new cell.**

13. Draw a diagram showing all the cells in the linked‑list representation of a buffer that contains the following text, with the cursor positioned as shown:

/Users/eroberts/Books/CS2-in-C++/Chapters/12-EfficiencyAndRepresentation/pictures/EditorDiagrams/HELLO3.png



14. Modify the diagram you drew in the preceding exercise to show what happens if you insert the character **X** at the cursor position.



15. What is meant by the phrase *traversing a list?*

***Traversing a list* refers to the process of looping through each cell in the linked list from beginning to end.**

16. What is the standard idiomatic pattern used in C++ to traverse a linked list?

for (Cell \*cp = start; cp != NULL; cp = cp->link)

17. Which editor operations require linear time in the linked‑list representation of the editor buffer? What makes those operations slow?

**The methods moveCursorBackward and moveCursorToEnd.**

18. What is a *time‑space tradeoff?*

**A *time‑space tradeoff* refers to situations in which you can improve running time by using more memory. In deciding whether to adopt such strategies, you have to decide whether the added time efficiency compensates for the space cost.**

19. What modification can you make to the linked‑list structure so that all six of the editor operations run in constant time?

**You can add a second set of links to the buffer so that each cell also points to the cell before it. Such a structure is called a *doubly linked list.***

20. What is the major drawback to the solution you offered in your answer to question 19? What might you do to improve the situation?

**Using a doubly linked list effectively doubles the space cost, since each character (one byte) requires two pointers (typically eight bytes) of link information. You can reduce the overhead by storing several characters in a single cell.**