Assignments - II

Write a program, called `MyNotePad`, that takes the following commands from the keyboard and performs the corresponding operations. If you are using C or C++, there should not be any memory leak. If you are using C or C++, do not use global variables; if you are using Java, do not use static variables.

1. `EXIT`: Terminates the program.

2. `PRINT`: Prints a string. We call this string `content`. When the program begins `content` is initialized as an empty string.

3. `PRINT_CURSOR`: Prints an integer. We call this integer `cursor`. When the program begins `cursor` is initialized as `0`. If the length of `content` is n, the value of `cursor` must be in $\{0, 1, 2, \ldots, n\}$.

4. `INSERT X`: Here, `X` is a character which is inserted at the position `cursor` of `content`. For example, if `content` is `"HLLO"`, `cursor` is 1, and the user provides the following command:
   `INSERT E`
   Then, `content` is updated to `"HELLO"` and `cursor` is updated to 2.

5. `SEEK X`: Here, `X` is an integer. If `X` $\in \{0, 1, 2, \ldots, n\}$, `cursor` is updated to `X`. Here, n is the length of `content`. If `X` is -1, cursor is updated to n. If, `X` $\notin \{-1, 0, 1, 2, \ldots, n\}$, the `cursor` does not move.

6. `DELETE X`: If `X` is 1, the character at position `cursor` of `content` gets deleted. If, `X` is -1, the character at position (`cursor` - 1) is deleted and `cursor` is set to (`cursor` - 1). If deletion is not possible, nothing changes. For instance, suppose `content` is `"WORLD"` and `cursor` is 3. Now, if the user provides the following command:
   `DELETE 1`
   Then, `content` is updated to `"WORD"`.
   Again, suppose `content` is `"WORLD"` and `cursor` is 3. Now, if the user provides the following command:
   `DELETE -1`
   Then, `content` is updated to `"WOLD"` and `cursor` is updated to 2.

7. `COPY X Y`: Here, `X` and `Y` are integers. This command copies the sub-string of `content` from index `X` to index `Y` and puts this sub-string in another string, called `clipboard`. `clipboard` is initialized as an empty string. If, `X`, `Y` $\notin \{0, 1, \ldots, n\}$, where n is the length of `content`, nothing changes.

8. `PRINT_CLIPBOARD`: Prints `clipboard`.

9. `CUT X Y`: Here, `X` and `Y` are integers. This command cuts (removes) the sub-string of `content` from index `X` to index `Y` and puts this sub-string in another string, called `clipboard`. If, `X`, `Y` $\notin \{0, 1, \ldots, n\}$, where n is the length of `content`, nothing changes. `cursor` is updated to n, where n is the length of `content` after modification.

10. `PASTE X`: Here, `X` $\in \{0, 1, \ldots, n\}$, where n is the length of `content`. This command pastes `clipboard` to position `X` of `content`. `cursor` is modified to `X` + m, where m is the length of `clipboard`.

11. UNDO: Performs an undo operation. This is applicable only to `INSERT`, `SEEK`, `DELETE`, `CUT`, and `PASTE`. Note that this operation can be applied multiple times. It erases the last change done to `content`, reverting it to an older state. Here `UNDO` will negate the last command done to `content` being edited.

12. REDO: Performs an redo operation. This is applicable only to `INSERT`, `SEEK`, `DELETE`, `CUT`, and `PASTE`. Note that redo can be applied multiple times. It reverses a previous `UNDO`. Here it repeats the previous action if it was something other than `UNDO`.

13. `SAVE FILENAME`: Here, `FILENAME` is a string indicating a filename. This command stores `content` to the file `FILENAME`.

14. `OPEN FILENAME`: Here, `FILENAME` is a string indicating a filename. This command sets `content` as the content of the file `FILENAME`. This command sets `cursor` to n, where n is the length of `content`. It makes the `clipboard` empty and resets/re-initializes any possible data structure required for performing `UNDO` and `REDO` operations.