

Computer Programming A, F  
FAST-NU, Lahore, Spring 2018

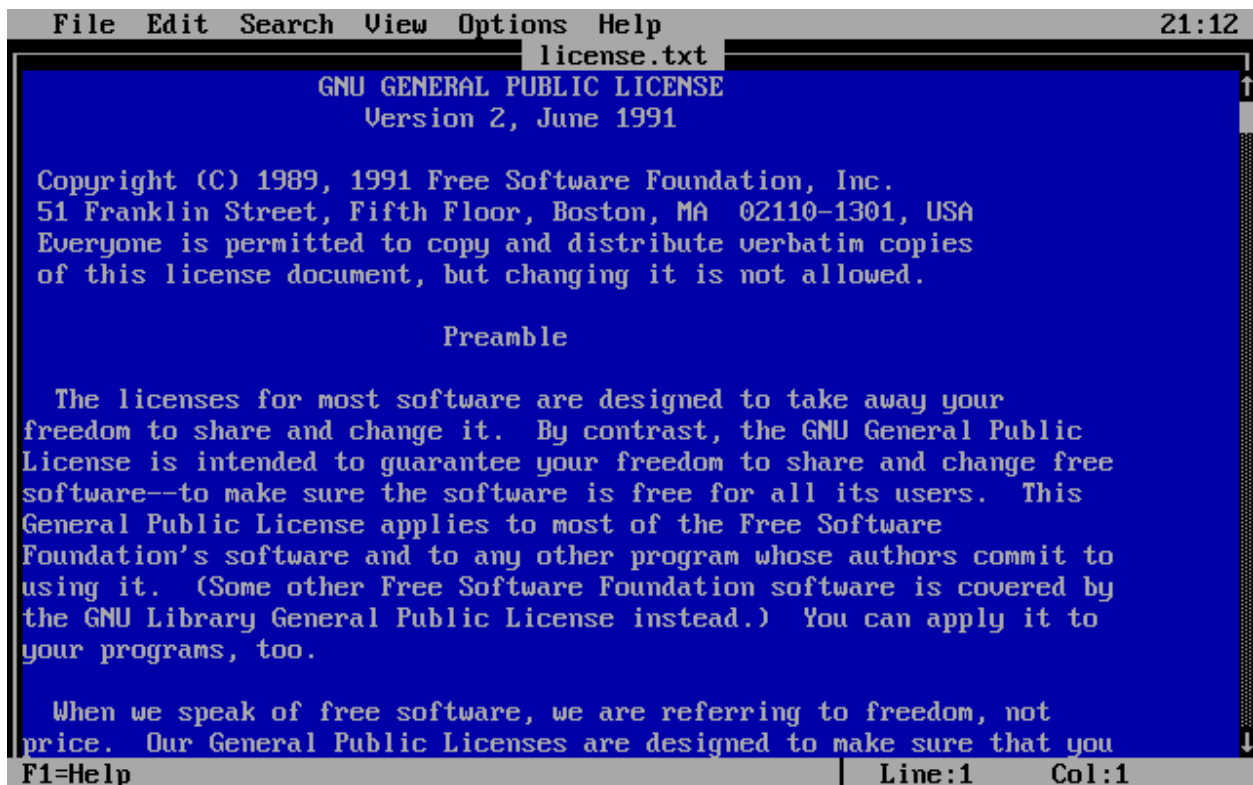
Homework 2

Text Editor Using Linked Lists and BGI

Due Friday March 09 11:55 P.M.

Marked out of 100 points.

The idea is to mimic the Edit program that used to be available on the DOS operating system. Following is a screenshot of that program:

A screenshot of a text editor window titled 'license.txt' showing the GNU General Public License, Version 2, June 1991. The window has a menu bar with 'File', 'Edit', 'Search', 'View', 'Options', and 'Help'. The text is displayed on a blue background with white characters. The license text includes the copyright notice for the Free Software Foundation, Inc., and the preamble explaining the purpose of the license. The status bar at the bottom shows 'F1=Help', 'Line:1', and 'Col:1'.

```
File Edit Search View Options Help 21:12
license.txt
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

                          Preamble

The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
F1=Help | Line:1 Col:1
```

You will use the BGI graphics (to be introduced in class on Thursday March 1<sup>st</sup>) in your visual editor to create something similar, even though your version will be much more limited.

**NOTE ON BGI with VS**

For students who want to start with graphics early, you don't have to wait till Thursday. Just read the simple procedure required to use BGI with visual studio here:

<https://www.cs.colorado.edu/~main/bgi/visual/>

I have tested some code with VS 2005, and that's worked fine.

Following is a slightly modified version of the main program given on the page. It uses the graphics.h library. The **initwindow** function opens a pixelated graphics window of specified dimensions. Here is an extensive list of the graphics functions available for use in BGI:

<https://www.cs.colorado.edu/~main/bgi/doc/>

You would need methods to highlight text, boxes, menus etc.

```
#include "graphics.h"

int main( )
{
    initwindow(400, 300, "First Sample");
    //circle(100, 50, 40);
    char ch=[2]={'\0', '\0'};
    while (true) //keep displaying
    {
        ch[0]=getch();//read the key-press
        outtext(ch);//display what was entered
    }
    return 0;
}
```

In your program you will maintain a doubly linked list – meaning each node has a next and previous pointer – called Text. Following is how a node in the list Text looks like:

```
struct textNode{
    char ch;
    textNode* next, *prev;
};
```

At the start of your program you will write a line like:

```
textNode * Text; //this will be the head of the list
```

In addition to this pointer you will need to maintain at all times a pointer called cursor (on screen this will appear as a short thick box or a blinking vertical bar):

```
textNode * cursor;
```

This pointer of course points to the current position of the cursor within the text. More precisely, it points to the node immediately behind the current position of the cursor.

The user should be able to perform at least the following functions:

(anything interesting you implement in addition to these features will be rewarded).

- When the user types a data input key such as an alphabet, number or symbol, a `TextNode` with the corresponding character is added immediately after the node pointed to by the cursor and the cursor is moved one step forward.
- When the user presses the backspace key the node pointed to by the cursor is removed and the cursor moves one step back.
- The user should be able to use the direction keys to move the cursor within the text without editing the text. Note: vertical direction keys will move cursor within lines.
- The user should be able to insert a new line wherever she wishes in text. Note that a new line is nothing but a character as far as the text is concerned, but it is displayed in a special way.
- Using the Shift + direction keys the user should be able to select the text. She should be able to use Ctrl+X, Ctrl+C and Ctrl+V to cut, copy and paste segments of text to any point of their choice within the text.
- User should be able to find Ctrl+F and replace Ctrl+R text. In this case a small window (maybe near the bottom of the screen) should ask the user for the text to be found and the new text to replace it.
- When the screen is filled with text the user should be able to keep writing and scroll up and down in the text. The program should paginate the text.
- User should be able to save and load the text to and from .txt files.

I leave the rest of the application open to your personal initiative and creativity. Graphically speaking, the text should be surrounded by a thick black border – like in the edit program – and if you're up for it you may add menus for the various options listed above, however, this will be for extra credit and your personal learning.

\*\*\*

THE END