# 24780 Engineering Computation: Problem Set 7

(*) In the following instructions (and in all course materials), substitute your Andrew ID wherever you see *yourAndrewId*.

You need to create a ZIP file (which may appear as a compressed folder in Windows) and submit the ZIP file via the 24-780 Canvas. The filename of the ZIP file must be:

```
PS07-YourAndrewID.zip
```

For example, if your Andrew account is hummingbird@andrew.cmu.edu, the filename must be:

```
PS07-hummingbird.zip
```

Failure to comply with this naming rule will result in an automatic 5% deduction from this assignment's credit. If we cannot identify the submitter of the file, an additional 5% credit will be lost. If we are ultimately unable to connect you with the submitted ZIP file, you will receive 0 points for this assignment. Therefore, ensure strict adherence to this naming rule before submitting a file.

The ZIP file must be submitted to the 24-780 Canvas. If you find a mistake in a previous submission, you can re-submit the ZIP file with no penalty as long as it's before the submission deadline.

Your Zip file should contain only two files, ps7.cpp and ps7.png. Do not include project files and intermediate files generated by the compiler. But, do not worry about some files or directories that are automatically added by the archiver (__MACOSX__ file for example).

Notice: The grade will be assigned to the final submission only. In the case of multiple file submissions, earlier versions will be discarded. Therefore, when resubmitting a ZIP file, it MUST include all the required files. Also, if your final version is submitted after the submission deadline, the late-submission policy will be applied, regardless of how early your earlier version was submitted.

**Ensure that your program can be compiled without errors on one of the compiler servers. Do not wait until the last minute, as the compiler servers may become very busy just minutes before the submission deadline!**

Submission Due: Please refer to Canvas.

START EARLY!

Unless you are a good programmer, there is no way to finish the assignment overnight.

## PS7 Bitmap Pattern Editor [ps7.cpp & screenshot of the compiler server showing no error ps7.png] (100 pts)

Make a program that allows the user to edit a bitmap pattern.

1. Make a class called CharBitmap that has three member variables:
   ```
   int wid,hei;
   char *pix;
   ```
2. Add a constructor so that the bitmap is empty (size is 0x0 and pix is nullptr) when created. (Or, you can do it in RAII style, too)
3. Add a member function called CleanUp, which deletes pix if a memory chunk is allocated, and make it 0x0 size, and nullify pix.
4. Add a destructor so that the bitmap is cleaned up when destroyed.
5. Add a member function called Create that takes two integer parameters w and h. You can assume that the calling function is responsible for giving correct w and h values. This function allocates an array of char long enough to store w*x elements, and also copy w and h to wid and hei members. Pixel values must be initialized to zero. Pretend that the class may be used by someone else in your project team. That person may call Create function twice in a row. This function must not cause memory leak in such a situation.
6. Add a member function called SetPixel which takes two integer parameters, x and y, and a char parameter p. Again, your teammate who uses SetPixel may give (x,y) that is out of the bound of the pixel. The program must not crash or write a value to unallocated space. (Do nothing if (x,y) is out of bound.) If (x,y) is within (0,0)-(wid-1,hei-1), set the pixel value at (x,y) as p.
7. Add a member function called GetPixel which takes two integer parameters, x and y, and returns the pixel value at (x,y). If (x,y) is out of bound, return 0. You know what the return type should be, right?
8. Add a member function called Draw. No parameters. This Draw function must magnify one pixel of the bitmap into 20x20 pixels on the window (can be drawn by GL_QUADS), and draw a pixel in the following color:
   ```
   Pixel Value 0 Black  (R=0, G=0, B=0)
   Pixel Value 1 Blue  (R=0, G=0, B=255)
   Pixel Value 2 Red  (R=255, G=0, B=0)
   Pixel Value 3 Magenta  (R=255, G=0, B=255)
   Pixel Value 4 Green  (R=0, G=255, B=0)
   Pixel Value 5 Cyan  (R=0, G=255, B=255)
   Pixel Value 6 Yellow  (R=255, G=255, B=0)
   Pixel Value 7 Blue  (R=255, G=255, B=255)
   ```
9. In main function, prompt the user for the dimension of the bitmap as:
   ```
   Enter Dimension>
   ```
   and get two integer values separated by a space from the console window. Use fgets or std::getline to take 1-line of input, and use parsing to get two values. If the user enters only

one value, or one of the dimensions is zero, or if one of the two values is greater than 64, prompt the user to enter the dimension again (repeat until the user enters a valid value).

10. The main function must have a variable of CharBitmap class. After the user specifies the bitmap dimension, create a bitmap of the specified dimension.
11. Open the window that is 20 times the bitmap dimension specified by the user (so that image drawn by the Draw function exactly fits on the window.) The program should continue to run until the user presses the ESC key.
12. Draw a lattice (vertical and horizontal lines) to show pixel boundaries with white lines. i.e., Vertical and horizontal must be separated by 20 pixels each.
13. In the main loop, call Draw member function of the bitmap and show it on the window. At the beginning, you are supposed to see all black pixels.
14. (14) When the user presses a key between FSKEY_0 and FSKEY_7, change the pixel value under the mouse cursor to the corresponding color. To get a color code, you can say:
    `int colorCode=key-FSKEY_0;`

Your program must not crash for any input from the user. Fig.1 shows a sample screenshot.

**2 points bonus: When the user presses S key, save the current bitmap in a text file called pattern.txt in the following format. First two numbers are width and height**

```
16 16
7770777777770777
7770077777700777
7770207777020777
7770270770720777
7770770000770777
7707777777777077
7707700770077077
7707777777777077
7077000770007707
7070414004140707
0770414004140770
0777000770007770
7077777777777707
7077777337777707
7700077777700077
7777755566677777
```

**3 points bonus (5 points total with the save-to-file): When the user presses L key, read the bitmap contents from pattern.txt. (You don't have to resize the window. If the bitmap does not fit in the window size, that's fine.)**

Test your program with the compiler server, take a screenshot of the compiler server showing no error, and include the screenshot in the zip file you submit.

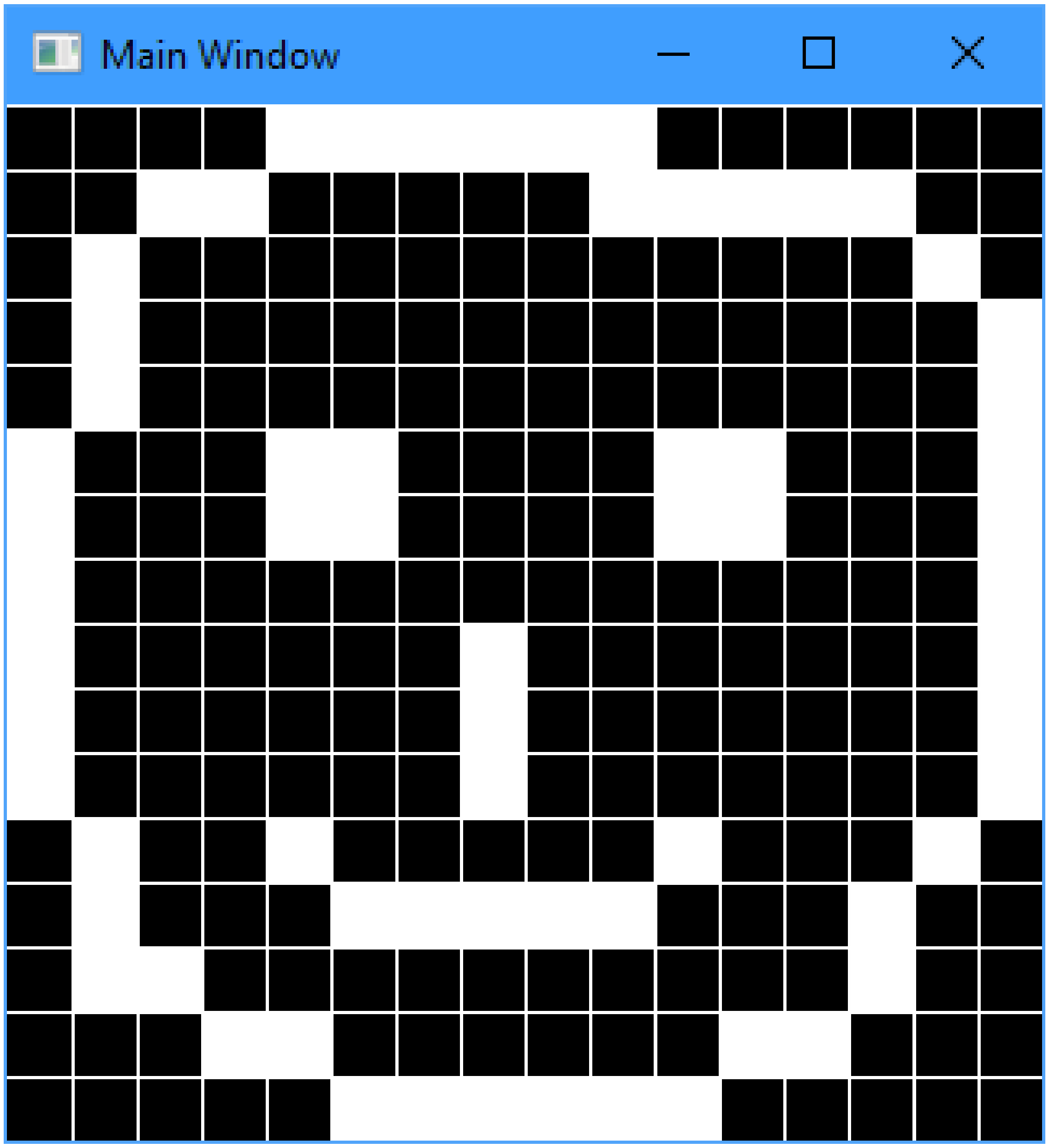

Fig. 1: Sample Screenshot

## Test Your Program with One of the Compiler Servers

Test your program with one of the following compiler servers:

```
http://freefood1.lan.local.cmu.edu
http://freefood2.lan.local.cmu.edu
http://freefood3.lan.local.cmu.edu
http://freefood4.lan.local.cmu.edu
```

You need to make sure you are not getting any errors (red lines) from the compiler server.

It is a good practice to remove warnings as well. However, we will not take points off for warnings as long as your program satisfies requirements of the assignment.

You can only access these servers from CMU network. If you need to access from your home, use CMU VPN. Please visit the CMU computing services web site how to install the VPN.