COMP0002 Programming Principles

Exercises 4

Purpose: Start by writing several drawing programs that use animation, to get familiar with how animation works ready for the coursework. Then write several programs that use pointers and dynamic memory, and try a couple of more complex programs if you have time.

Goal: Complete as many of the exercise questions as you can. If you are keeping up, you need to do at least the core questions. The additional questions are more challenging and are designed to stretch the more confident programmers. Don't worry if you can't do them now, but be prepared to come back and try them later on. If you run out of questions then try extending the questions here or finding your own.

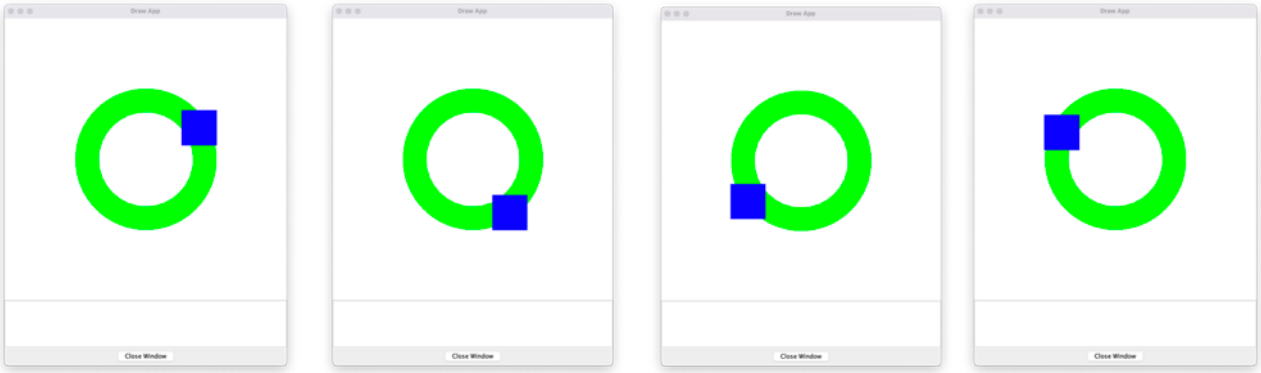
Feedback: It is important that you get feedback on your exercise answers so that you know they are correct, that you are not making common mistakes, that the program code is properly presented and that you are confident you have solved the problem properly. To do this, get your answers reviewed during a lab or mentor session.

NOTE: Keep a copy of all your exercise answers.

Core Questions

Q4.1

a) Write an animated drawing program to display an animation that looks like this (also see the video of this program on Moodle):



The green circular shape is on the background and doesn't move. The blue square moves following a square path over the background shape.

b) Write a second version that moves the square along a circular path.

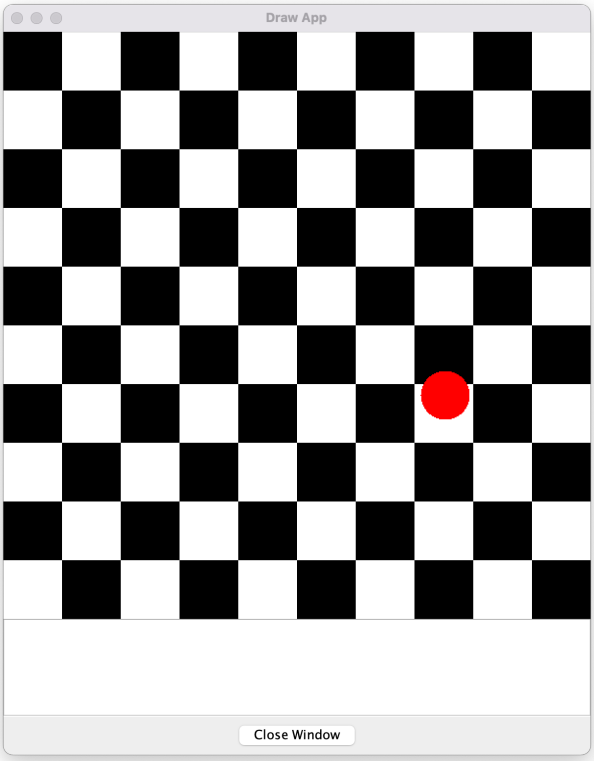
c) Write a third version where the square moves following a path specified in a series of coordinates stored in an array. The coordinates could be read from a data file.

d) Write a final version where the square also slowly rotates as it moves.

Q4.2

a)

Write an animated drawing program to display a bouncing ball with a checkerboard background that looks like this (also see the video on Moodle):



The ball bounces off the edges of the window, at a random angle.

b) Write a version that has multiple bouncing balls of different sizes and colours. Provide a different background if you like.

Q4.3 Write a version of the string copy function that takes a single string argument and creates a new copy in dynamic (heap) memory using malloc. The function signature should be:

```
char* stringCopy(char *s)
```

Q4.4 Write a function with the signature `int strend(char *s, char *t)` that returns true (1) if the string pointed to by t occurs at the end of the string pointed to by s. Otherwise the function returns false (0).

Q4.5 Write a function with the signature `int* sort(int* n)` that takes a pointer to an array of 10 integers, and returns a pointer to a *new array* containing the integers in sorted order.

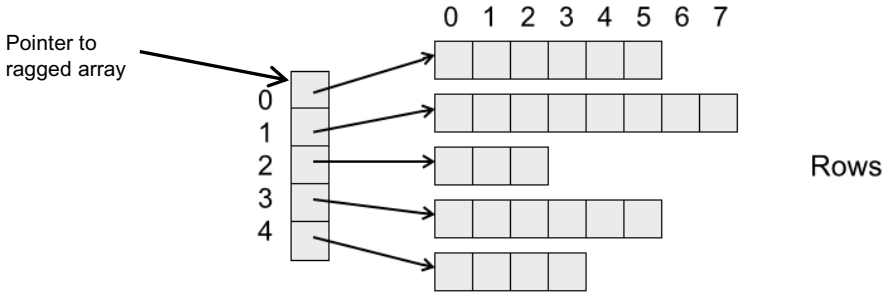
Note that a sort function with the signature above is not given the size of the array to be sorted, and can only work with a fixed size array where the size is specified elsewhere in the program. A better signature would be `int* sort(int* n, int size)` that adds a second parameter

giving the size of the array to be sorted. Modify a copy of your function, edit it to have this signature with a new name, and update the function body to use the size parameter.

Challenge Questions

Q4.6

a) Write a function to create and return a pointer to a ragged 2D array data structure. A ragged 2D array has rows of different lengths. The basic data structure has a pointer to an array of pointers that point to each row, each of which is an array of integers:



The function should take as a parameter the number of rows and an array that specifies the length of each row. The pointer to the ragged array will be of type int**, a pointer to an array of int pointers. All the arrays should be allocated memory dynamically on the heap using malloc.

Write a second function that frees all the memory used by a ragged array, and some other functions to confirm that creating and printing your ragged arrays works correctly.

b) It would be better if the ragged array data structure also stored information about the number of rows and the length of each row. Modify your data structure so that it can store the size information, think about using a struct for this. Provide updated functions to create and free a ragged array, along with functions to get and set values in a ragged array and print out a ragged array.

Q4.7 Write a program that takes several paragraphs of text and formats the text into two columns, so that the text can be printed on the screen in the two columns. Each column should be thirty characters wide and text should be adjusted so that only complete words show on each line. Make use of pointers, strings (character arrays), with all memory allocated dynamically on the heap.

Additional challenges:

i) Read the original text from a file.

ii) Write the formatted text out to a file.

iii) Allow the number and width of columns to be specified.

iv) Add support for hyphenation, allowing words to be split across two lines.