

CSC3002: Introduction to Computer Science

Assignment 4

There are FOUR questions included in Assignment 4. Please find the zip file of the QT project for Assignment 4 in the BB system. Please refer to the test program in ".cpp" for each question to implement your programs and test them accordingly.

Question 1 & 2

Design and implement a class called **StrArray** (a dynamic array of `std::string`) that implements the following methods:

Part 1:

- The default constructor is already given.
- A destructor that frees any allocated heap storage allocated.
- A method **get(k)** that returns the element at position `k`. If `k` is outside the vector bounds, **get** should call **error** with an appropriate message.
- A method **set(k, value)** that assigns `value` to the element at position `k`. As with **get**, the **set** method should call **error** if `k` is out of bounds.

Part 2:

- A method **grow()** to increase the capacity of the array
- A method **push_back()** to add a new element to the array;
- A method **operator[](n)** to get the element at the given index `n`.

You must make sure that `string`'s constructor and destructors are called correctly and the memory usage is correct.

Two sample tests are provided in the code.

Very important hints:

- You should use `::operator new(size)` to allocate an area of uninitialized memory.
- You should use `::operator delete(ptr)` to deallocate the memory area created by `::operator new`
- To construct an element in a uninitialized memory area, you should use placement new:
`new (pointer_to_position) std::string (value);`
- You should use `std::destroy` to destroy elements within the memory area without deallocating the memory.
- To move elements from original place to a new uninitialized area, you can use `std::uninitialized_move(original_start, original_end, new_start);`

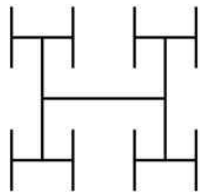
- The above functions are included in <new> and <memory>

Question 3

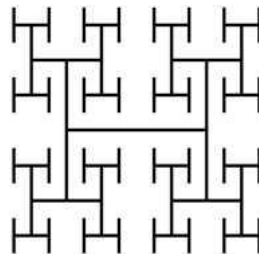
If you search the web for fractal designs, you will find many intricate wonders beyond the Koch snowflake illustrated in this chapter. One is the H-fractal, in which the repeated pattern is shaped like an elongated letter H that fits inside a square. Thus, the order-0 H-fractal looks like this:



To create the order-1 fractal, all you do is add four new H-fractals—each half the original size—at each open end of the order-0 fractal, like this:



To create the order-2 fractal, you just add even smaller H-fractals (again half the size of the fractal to which they connect) to each of the open endpoints. This process gives rise to the following order-2 fractal:



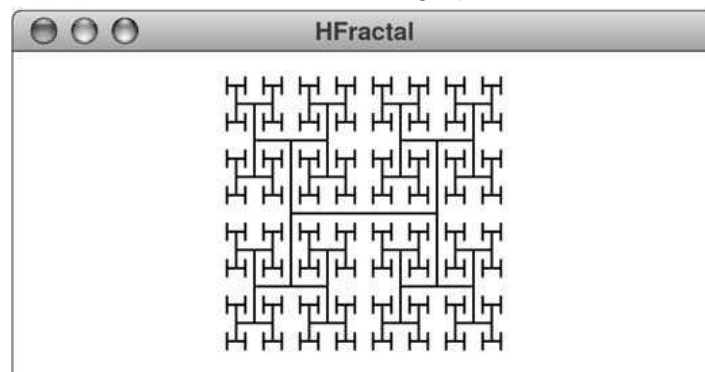
Write a recursive function

drawHFractal(GWindow & gw, double x, double y, double size, int order);

where **x** and **y** are the coordinates of the center of the H-fractal, **size** specifies the width and the height, and order indicates the **order** of the fractal. As an example, the main program

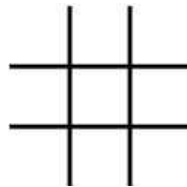
```
int main() {
    GWindow gw;
    double xc = gw.getWidth() / 2;
    double yc = gw.getHeight() / 2;
    drawHFractal(gw, xc, yc, 100, 3);
    return 0;
}
```

would draw an order-3 H-fractal at the center of the graphics window, like this:

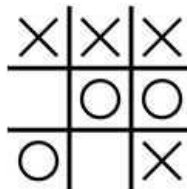


Question 4

The game of **tic-tac-toe** (or **naughts and crosses**) is played by two players who take turns placing **Xs** and **Os** in a 3×3 grid that looks like this:



The object of the game is to line up three of your own symbols in a row, horizontally, vertically, or diagonally. In the following game, for example, **X** has won the game by completing three in a row across the top:

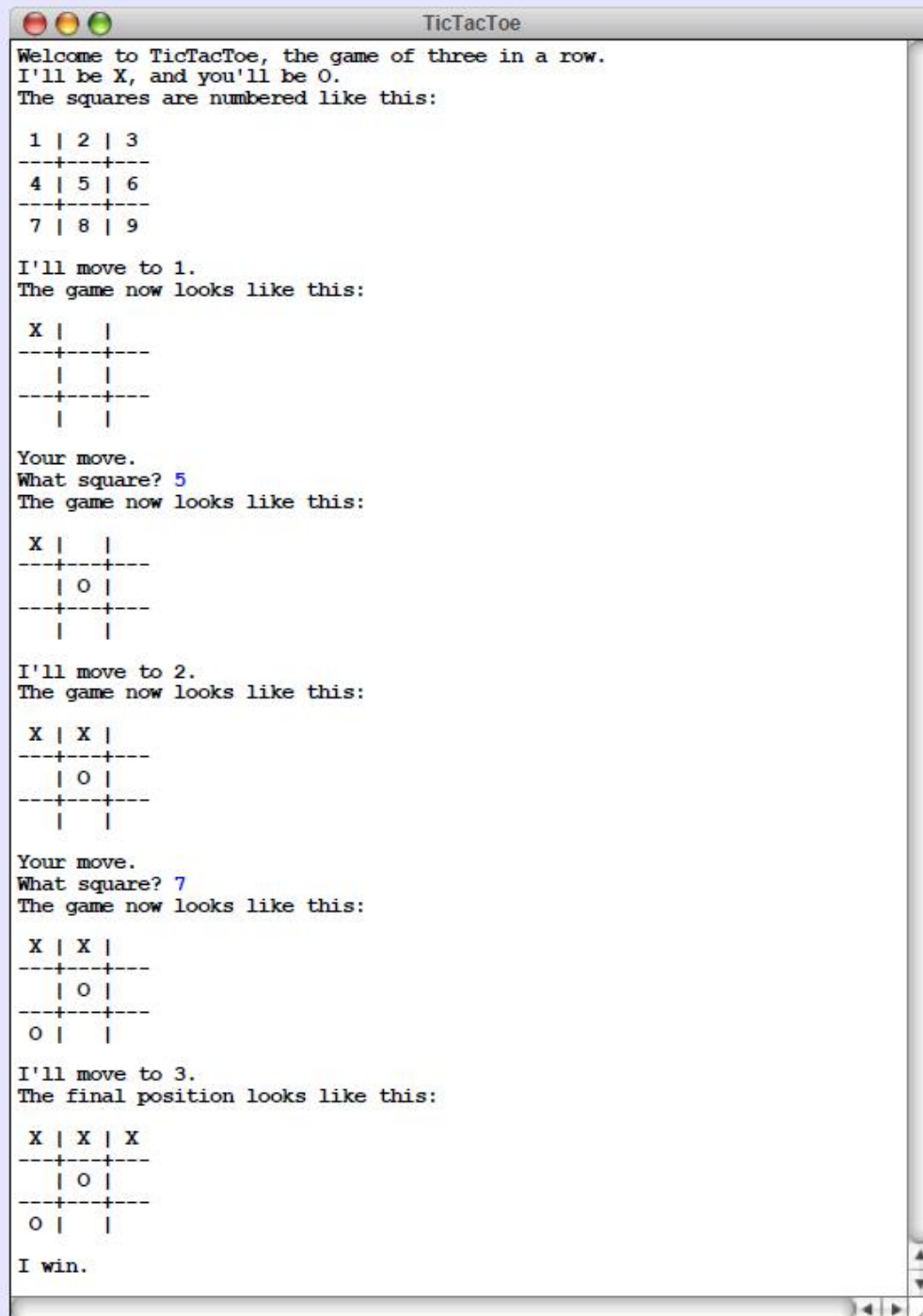


If the board fills up without anyone completing a row, the game is a draw, which is called a **cat's game** in tic-tac-toe.

Write a program that uses the minimax algorithm to play a perfect game of tic-tac-toe. Figure 1 shows a sample run against a particularly inept player.

Notice this is a small project, you can reuse the definitions in the given template or write a new one; Make sure to fill the the PROVIDED_TEST part in `tictactoe.cpp` so that we can start your game.

FIGURE 1 Sample run of the tic-tac-toe game



```
TicTacToe
Welcome to TicTacToe, the game of three in a row.
I'll be X, and you'll be O.
The squares are numbered like this:

 1 | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9

I'll move to 1.
The game now looks like this:

X |  | 
---+---+---
  |  | 
---+---+---
  |  | 

Your move.
What square? 5
The game now looks like this:

X |  | 
---+---+---
  | O | 
---+---+---
  |  | 

I'll move to 2.
The game now looks like this:

X | X | 
---+---+---
  | O | 
---+---+---
  |  | 

Your move.
What square? 7
The game now looks like this:

X | X | 
---+---+---
  | O | 
---+---+---
O |  | 

I'll move to 3.
The final position looks like this:

X | X | X
---+---+---
  | O | 
---+---+---
O |  | 

I win.
```