The aim of this report is to show my progress on the first Programming Project and to demonstrate my understanding of the code I wrote. I am going to write about each of the tasks individually and attach screenshots where necessary. In the end I am also going to reflect on those aspects of the Project that I found most difficult and the things I could have done better.

**Task 1: Destroy the tree**

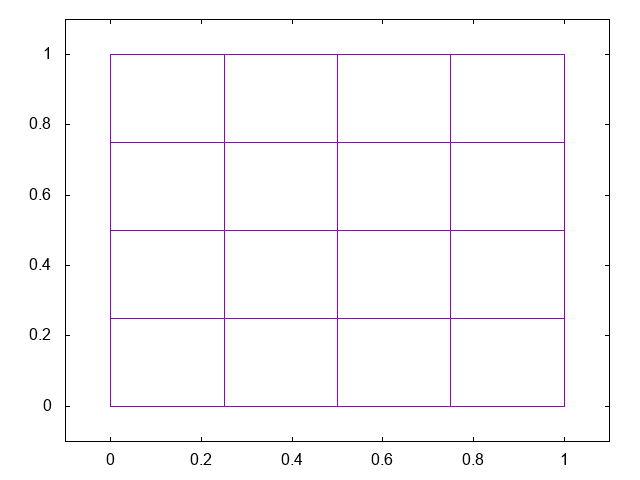
It is of prime significance to free all the memory which has been allocated. To ensure that there are no memory leaks the tree has to be destroyed (recursively).

1. A full tree at Level 2  
   Expected results: Creation of a full tree of level 2, memory leaks when the memory isn’t freed, no memory leaks after destroying the tree.

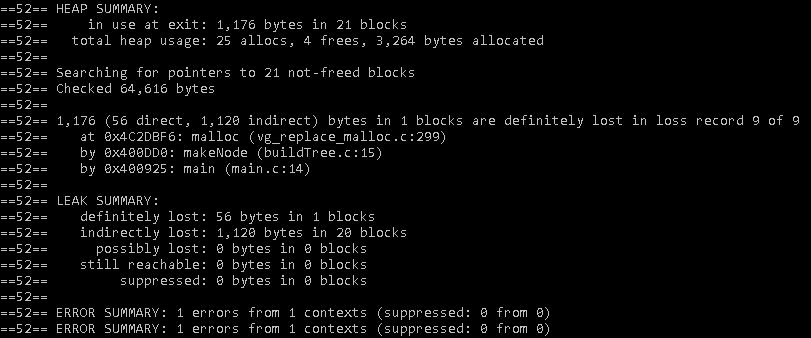
Expected input:

1  
1  
Final results:

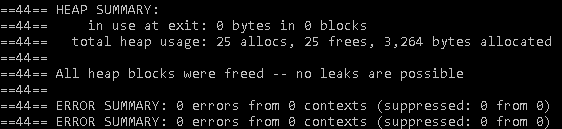
A full level 2 tree.



Before destroying the tree



After destroying the tree



1. A non-uniform Level 3 tree that you define.

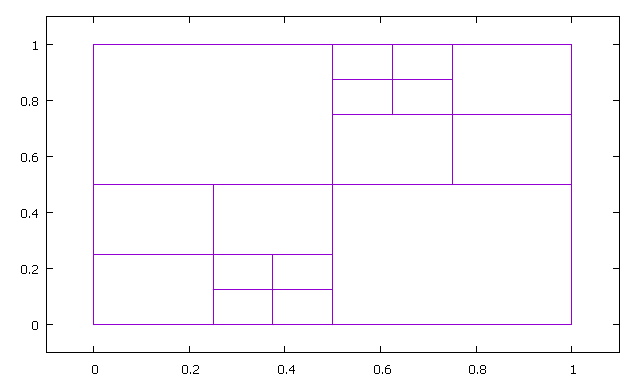
Expected results: Creation of a non-uniform tree of Level 3, memory leaks when the memory isn’t freed, no memory leaks after destroying the tree.  
Expected input:

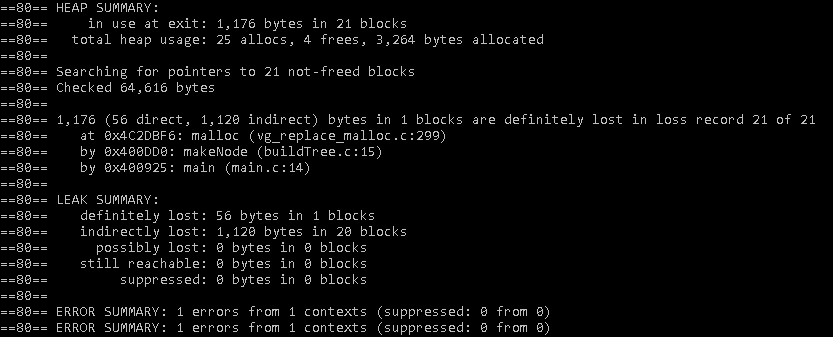
1

2

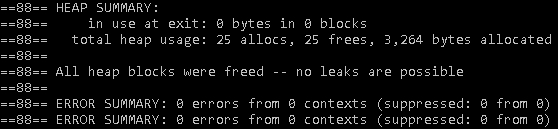
Final results:

A level 3 tree



Before destroying the tree

After destroying the tree:



Code

Checks if the node is a leaf node and frees the allocated memory if it is. Otherwise, the recursive function goes through all 4 children of the node until a leaf is reached. After freeing the children, the memory allocated to the parent is also freed.

**void** destroyTree(Node \*node) {  
 **int** i;  
  
 **if**( node->child[0] == NULL )  
 free(node);

**else** {  
 **for** ( i=0; i<4; ++i ) {  
 destroyTree( node->child[i] );  
 }  
 free(node);  
 }  
}

### Task 2: Growing the Quadtree

This function recursively grows the tree uniformly overall by one level.

1. Specify a test for this function with an initial tree structure of a full Level 2 tree, and the expected results.

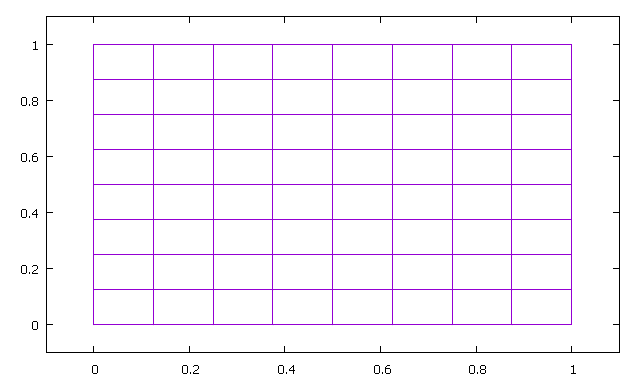
Expected results: Creation of a full Level 3 tree

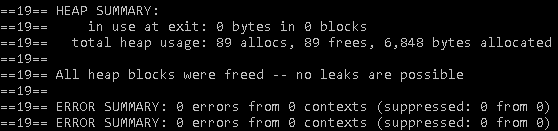
Expected input:

2

1

Final results:





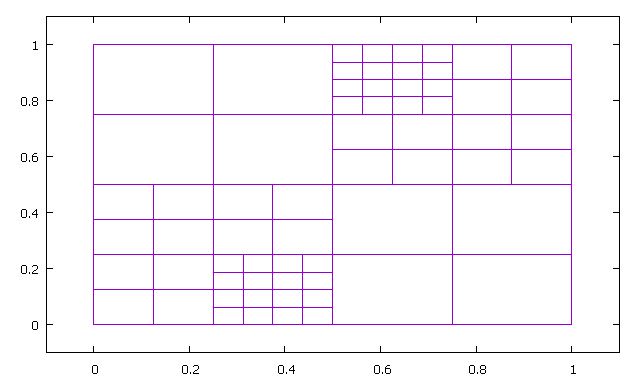
1. Specify a test for this function with a non-uniform Level 3 tree structure that is not full, and the expected results.   
   Expected results: Creating a non-uniform level 4 tree from the level 3 tree in task 1

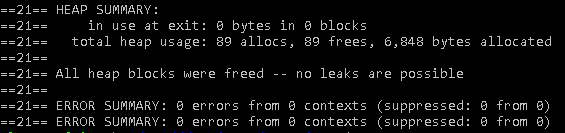
Expected input:

2

2

Final results:





Code:

This function recursively makes children for all the leaf nodes. If the node is a parent, the function goes through all its children and adds children when a leaf node is reached.

**void** growTree( Node \*node ) {  
 **int** i;  
  
 **if**( node->child[0] == NULL) {  
 makeChildren(node);  
 }  
  
 **else** {  
 **for** ( i=0; i<4; ++i ) {  
 growTree (node->child[i]);  
 }  
 }

### Task 3: A limit on tree level

(i) Making some basic assumptions about the size of primitive data types estimate the memory use for a tree Node on paper.

Explain your answer and assumptions in the report.

The Node consists of 1 int, an array of 2 doubles and pointers to 4 children (also nodes). Judging by the size of primitive datatypes, 1 node should be 4 bytes (int) + 2\*8 bytes (doubles) + 4\*8 bytes (pointers) it should be 52 bytes.

(ii) Produce a table showing how much memory you would expect to use for a full tree with 5, 6, 7, 8, 9 and 10 levels.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| levels | 5 | 6 | 7 | 8 | 9 | 10 |
| size (bytes) | 70,980 | 283,972 | 1,135,940 | 4,543,812 | 18,175,300 | 72,701,252 |

1. Using your quadtree implementation and running the code with valgrind produce data on the actual memory use of your code for a full tree with 5,6,7,8 levels.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| levels | 5 | 6 | 7 | 8 |
| size (bytes) | 76,440 | 305,816 | 1,223,320 | 4,893,336 |

It turns out that 1 Node is actually 56 bytes which is why there is a difference between the expected and the actual size of the quadtrees. The compiler adds 4 bytes of padding because of the data alignment requirements of the system. (https://stackoverflow.com) In other words it has to be a multiple of 64 bits or 8 bytes. (52 isn’t, but 56 is).

1. If you would like to limit the overall memory use of the application to 20Mb what maximum level should you choose?

20 Megabits – level 7 since it’s 9Mb and 8 is over 37Mb

20 Megabytes – level 9 since it’s 18.6MB and 10 is over 72MB

1. Implement the maximum tree level as a parameter in your code so that the tree cannot grow beyond that level. Your code should continue execution but the tree is prevented from growing beyond the maximum level.

Use your growTree() function with

1. maximum level=3

(ii) maximum level=4