1310 PROGRAM 2 – Linked List / Search / Sort

Dinosaur Fight Game



# Files to Be Included In Your Submission

* **LinkedList.h** – header file containing your LinkedList class. You can put the entire class & function definitions in this one file if you want. In fact, if you do the extra credit and make it in a Template class, I highly recommend keeping it all in this one file.
* **Dinosaur.h** – Dinosaur class specification file – you may also define your Dinosaur member functions here or separate them out in an implementation file like I did
* **[optional]** **Dinosaur.cpp** – Dinosaur class implementation file
* **DinosaurFight.cpp** – contains the main function and any additional functions you created to make the program run
* **Makefile** (provided for you) – if you name your files differently or have more files, then you will need to modify the makefile.
* **dinosaurFile.txt** (provided for you) – feel free to add to the file your own dinosaurs

# Description

You are creating a game where players can pick their dinosaur and then the dinosaurs fight each other.

Each dinosaur has a **life** point value and a **hit** point value.

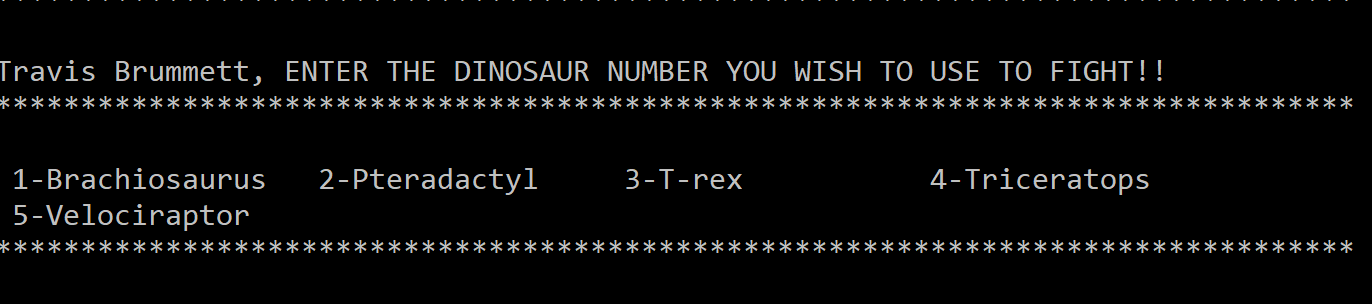
On a player’s turn, they roll one six-sided die and that die value is multiplied by the dinosaur’s hit points. That is the amount of damage that is done to the other player’s dinosaur’s life points.

# SPECIFICATIONS

The program should be created in a source file named **DinosaurFight.cpp**. Below is the logic of the main function.

**You may separate the logic in the main function into as many other programmer-defined functions as you want and name them whatever you want.**

**However, your program must work as specified below:**

1. Create a linked list of **Dinosaurs** (**Dinosaurs.h** & **Dinosaurs.cpp**). Use the **LinkedList** class (**LinkedList.h**) to accomplish this.
2. Enter all the dinosaur data from file named **dinosaurFile.txt**. **Do not rename this text file** and **do not make the user enter in the filename** – just open dinosaurFile.txt and read in the data. Using the data in this file.
   1. After reading in the information about one dinosaur from the file, do a **Binary Search (RECURSIVE VERSION)** on the name of the dinosaur to see if a dinosaur with this same name already exists. If it does, then print that this dinosaur (print the name) was skipped because it is already in the dinosaur list.
   2. If the dinosaur does not already exist in the list, then
      1. create a new **Dinosaur** object,
      2. append the dinosaur to the **LinkedList** object,
      3. write and use the **MergeSort algorithm** to sort the nodes (**Dinosaurs**) in the **LinkedList** object by **Dinosaur name**.
      4. print the name of the dinosaur and tell the user that the dinosaur has been added to the list.
   3. After repeating this for all the dinosaurs in the text file, print a message to the user telling them how many dinosaurs were added to the list from dinosaurFile.txt.
3. Now allow the users to play the game. Allow **two** players to enter in their name and then use their names throughout the rest of the output.
4. Ask the users if they want to print out the detailed information about each dinosaur before having to choose which one they will use. Make sure to validate the user’s output for data type and correct possible values.
5. If the users answer yes, then print out all details about ALL dinosaurs.
6. Now allow player one to enter the dinosaur number of the dinosaur they wish to choose. Print out the dinosaur’s names like the example below in four columns and each dinosaur should have a number beside it so the user can pick the dinosaur with the number instead of having to type in the name.  
   
7. Then allow player two to pick a dinosaur. They can pick the same dinosaur if they want.
8. Now the players are ready to play. There should be **three** rounds, unless one of the dinosaurs has <= 0 life points left.
9. For each round, give each player one turn, starting with player 1. Below is what should happen for each player on their turn  
   1. Tell the player to hit enter to roll the die.
   2. Generate a random number between 1 & 6
   3. Multiply that random number by the player’s HIT points.
   4. Subtract that amount from the opposing player’s dinosaur’s LIFE points.
   5. If the opposing player’s dinosaur’s LIFE points are now <= 0, then this player won the game (tell them)
10. If the game progresses through all three rounds, then after the third round is over see which dinosaur has the most life points and then announce that player (and dinosaur) as the winner.
11. Ask the user if they want to play again. If so, then you should start over with asking the player’s for their names.

## dinosaur Class

### Attributes

* Dinosaur’s name (string)
* Dinosaur’s description (string)
* Two integers – one for life points and one for hit points

### Member Functions

* Constructor – initialize all attributes to values sent to constructor
* Accessor & mutator functions for all attributes
* Print dinosaur – a function to print the details of a dinosaur out to the screen in a nice, readable way

## linked list Class

**EXTRA CREDIT OPPORTUNITY:**

Create the **LinkedList** class as a **template class** for 5 extra credit points!!!

### private memebers

* Create a structure called ListNode, which should hold a Dinosaur and a pointer to the next ListNode
* ListNode pointer called head – will eventually point to the first node in the linked list
* ListNode pointer called tail – will eventually point to the last node in the linked list
* Integer called numNodes – will hold the number of nodes in the linked list

### public members

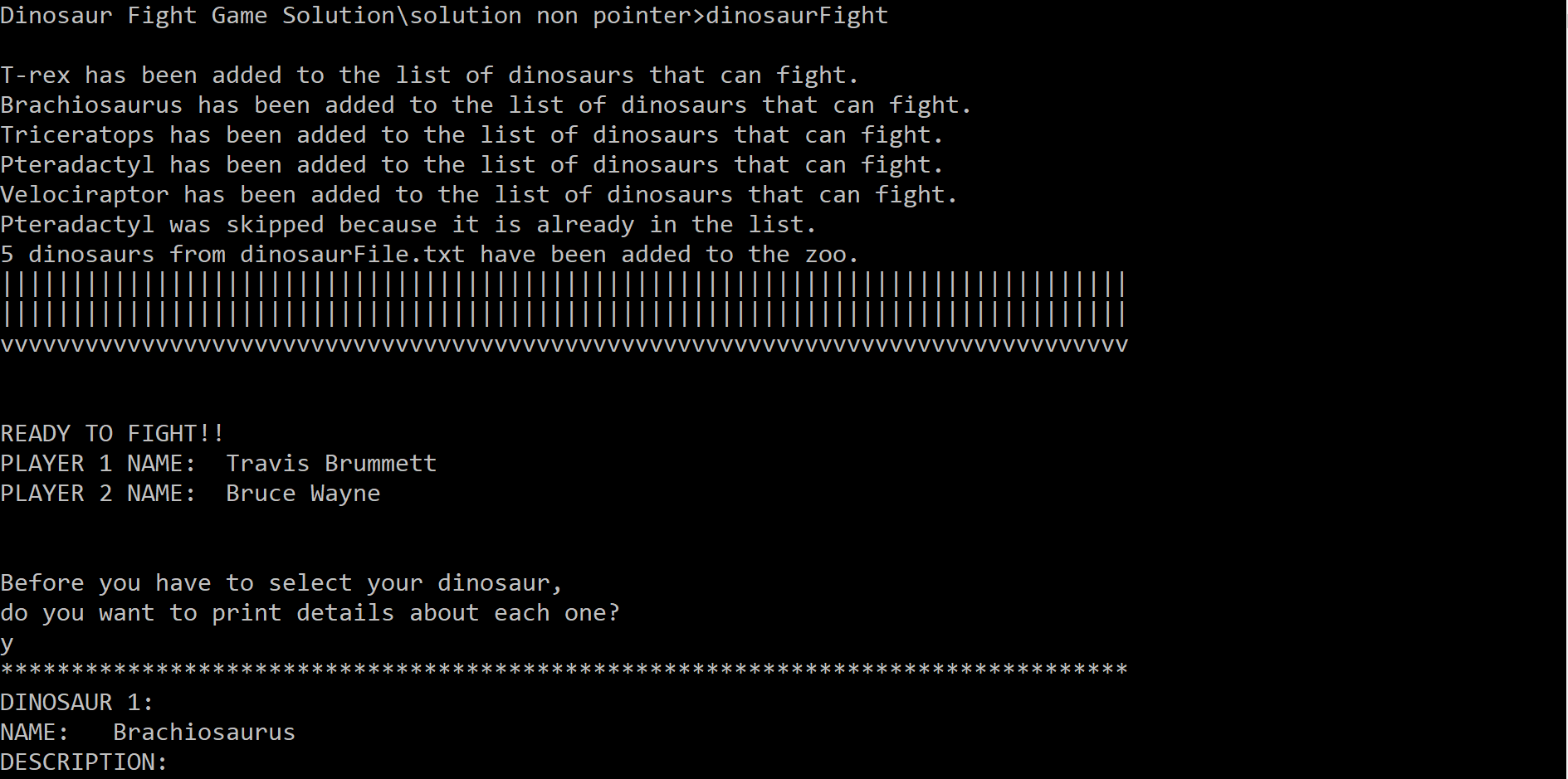
* **Constructor** – initialize head, tail, & numNodes
* **Destructor** – delete all nodes in the list
* **getLength** accessor
* **getNodeValue** accessor – should return a Dinosaur
* **appendNode** – appends a node containing the value passed into nodeValue, to the end of the list
* **insertNode** – this function is passed a node value (Dinosaur) and a position where the node should be inserted. The first node is position 0.
* **deleteNode** – finds the node with the argument position & deletes it

# How TO Turn in your Program

* Zip all the files required to compile & run your program – include the **Makefile** & **dinosaurFile.txt**. You may modify dinosaurFile.txt if you want to contain different dinosaurs.
* Upload it to the PROGRAM 2 submission folder in ilearn by the due date.

# Sample Output ONE – showing whole program

The text that is highlighted in yellow indicate user input.



**Notice that because of the binary search, Pteradactyl was skipped.**

