UNIVERSITY OF THE WEST INDIES, CAVE HILL CAMPUS

Department of Computer Science, Mathematics & Physics

COMP1210 Computing 2

Due Date: May 14nd 2021 Midnight (HARD DEADLINE)

This assignment is worth 20% of your coursework

PART TWO

The focus of this assignment is to simulate a habitat for which the GreenMonkey objects from assignment one will live and interact. The habitat area will be presented as an n x n grid, where each cell of the grid holds a **Linked List** of GreenMonkey objects. Below is the code that defines the **GreenMonkey_LinkedList** and the **Habitat** classes. These two definitions along with the **GreenMonkey** class definition from assignment one are all placed in a file called **Classes.h**. All of your definition code for all three classes should be placed in a file name **Classes.cpp.**

Note: Please start a new project for this part of the assignment do not try to simply add this work into the project for part one.

```
class GreenMonkey_LinkedList {
private:
	GreenMonkey* head;
	int idCount;
public:
	GreenMonkey_LinkedList();
	void printList();
	void insert_at_start();
	void deleteNode(int)
	void set_idCount();
	int get_idCount();
	void update_id();
	GreenMonkey* get_Head();
	GreenMonkey* get_Node(int);
};
```

Specifications:

The functionality of the member functions of the **GreenMonkey_LinkedList** class:

- **GreenMonkey_LinkedList()** – this default constructor simply sets the **head** data member to NULL and the **idCount** data member to zero. [2 Marks]

- **printList()** - this member function invokes the **print** member function of all the nodes in the linked list. [5 Marks]

- **insert_at_start()** - this member function adds a new GreenMonkey object to the Linked List. With this function you will invoke the **set_idCount()** member function which increments the **idCount** data member of the Linked List by 1. Also in this member function you should invoke the **set_ID()** member function and pass to it the **idCount** data member. [5 Marks]

- **deleteNode()** – this member function removes a node from the Linked List. The node that will be removed is the one that its ID value is equal to the value passed into this function as a parameter. [5 Marks]

- **set_idCount()** – this member function increments the **idCount** data member by one. [2 Marks]

- **get_idCount()** – this member function returns the **idCount** data member. [2 Marks]

- **update_id() –** this member function does two task, firstly it scans the Linked List and sets the **ID** data member of each node in the Linked List, i.e the first node **ID** value would be set to 1 and all subsequence nodes would be set to 1 more than the previous one. Secondly, the **idCount** data member of the **GreenMonkey_LinkedList** class will be set to the number of nodes in the Linked List. [5 Marks]

- **get_Head() –** this member function return the **head** pointer data member. [2 Marks]

- **get_Node()** – this member function accepts an integer parameter and it scans the Linked List searching for a node that has an ID value the same as the parameter passed into the function, if the a match is found this returns the matching node. [5 Marks]

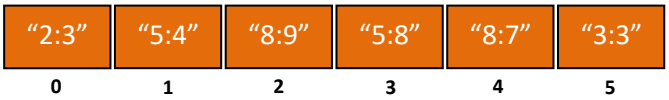
```
class Habitat
{
private:
	static const int RowCol = 10;
	GreenMonkey_LinkedList grid[RowCol][RowCol] = {};//creates a 10 X 10
grid, each grid cell will have an empty GreenMonkey LinkedList

public:
	vector<string> getEmptyCells();
	void intialization(int);
	void showGrid();
	void impact(GreenMonkey_LinkedList&, GreenMonkey*); //picks a random
GreenMonkey from the LinkedList in the 1st parameter and engage with the
GreenMonkey which is the second parameter.
	void invasion(int);
	void ageAll();
	void feedAll();
	void printAll();
	void getStatistics();
};
```

Specifications:

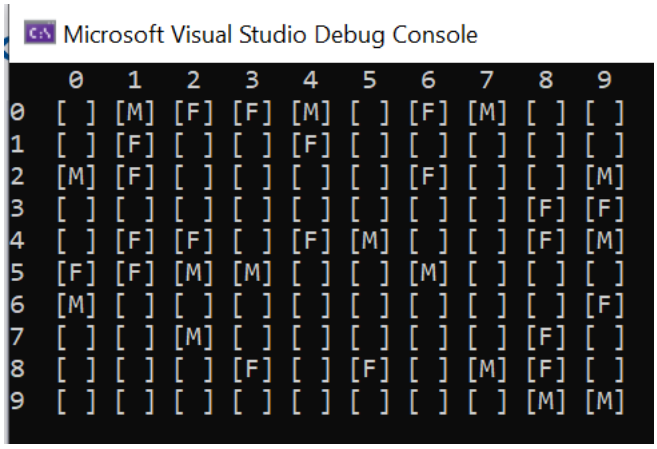
The functionality of the member functions of the **habitat** class:

- **getEmptyCells()** – this function scans the **grid** array and produces a vector that contains the (x, y) locations of all the empty cells in the **grid**. For example, say we have the following empty cells in our grid, (2,3) (5,4) (8,9) (5,8) (8,7) (3,3), our vector would look as follows: [5 Marks]



- **initialization(**int**)** – this function is use to initialize the grid with GreenMonkey objects, it takes an integer parameter which is the number of GreenMonkey objects to be placed in the grid. At the initialization phrase, GreenMonkey objects should only be placed in a grid cell that is empty, this is achieved by using the **getEmptyCells()** function from above. The GreenMonkey objects should be place randomly throughout the grid. [5 Marks]

- **showGrid()** – this function scans the grid and displays the gender of the GreenMonkey object that is at the head of the Linked List of an occupied grid cell. Below shows an example of the output from this function where the grid was initialized with 35 GreenMonkeys. [5 Marks]



- **impact(**GreenMonkey_LinkedList&, GreenMonkey***)**– this function accepts two parameters, a linked list of GreenMonkey objects and a single GreenMonkey object.
  It picks an object from the linked list at random, if the picked monkey is dead then nothing is done, otherwise if the gender of the picked monkey object is the same as the gender of the object in the second parameter, then the object in the second parameter will invoke its **fight** function passing in the picked monkey object.
  If the gender of the picked object is not the same as the gender of the object from the second parameter, then the two monkeys will mate, resulting in the creation of a new monkey object which will be added to the linked list. [5 Marks]

- **invasion(**int**) –** this function accepts one integer parameter which is the number of monkeys that will be invading the habitat area. A grid cell location is picked at random, if the cell is empty then a new monkey object is simply added to the empty linked list at the location, however if the linked list at the picked location is not empty then the new monkey object is added to the list and the **impact** function is invoked while passing the list at the location and the new monkey as its parameters. [5 Marks]

- **ageAll** – this function invokes the **age_me** function on all the monkeys in the habitat area. [3 Marks]

- **feedAll** – this function invokes the **eat** function on all the monkeys in the habitat area. [3 Marks]

- **printAll** – this function invokes the **print** function on all the monkeys in the habitat area, producing an output similar to the following: [10 Marks]

```
Cell->0:0 has 7 monkeys.
Name            Age  Weight    Gender    Status    Genes     Injured
=============================================================================
Duke            4    3.69      M         Dead      Mutant       1
Pepper          11   9.64      F         Alive     Normal       0
Simon           13   43.89     M         Alive     Normal       0
Tasha           11   30.52     F         Dead      Normal       0
Simon           12   39.86     M         Alive     Normal       0
Misty           12   10.22     F         Dead      Normal       1
Cleo            11   11.34     F         Alive     Normal       0


Cell->0:1 has 3 monkeys.
Name            Age  Weight    Gender    Status    Genes     Injured
=============================================================================
Riley           11   15.61     M         Dead      Normal       0
Pug             11   44.75     M         Alive     Normal       0
Lucky           14   8.87      M         Alive     Normal       2


Cell->0:2 has 7 monkeys.
Name            Age  Weight    Gender    Status    Genes     Injured
=============================================================================
Taz             4    2.65      M         Dead      Mutant       0
Beau            13   24.79     M         Alive     Normal       1
Jake            12   9.80      M         Alive     Normal       0
Oliver          10   18.92     M         Dead      Normal       1
George          5    3.24      M         Dead      Mutant       0
Sam             13   41.29     M         Alive     Normal       0
```

- **getStatistics** – this function displays the following facts:

    - Total monkeys in the habitat
    - Total male monkeys in the habitat
    - Total female monkeys in the habitat
    - Total alive monkeys in the habitat
    - Total dead monkeys in the habitat
    - Total mutant monkeys in the habitat

- Total normal monkeys in the habitat
- Average weight of all alive monkeys in the habitat
- Average age of all alive monkeys in the habitat
- Average injuries of alive monkeys in the habitat [10 Marks]

Below shows how and what the **getStatistics** member function should display.

```
================================================================
Description                                              Value
================================================================
Total monkeys in the habitat                               496
Total male monkeys in the habitat                          250
Total female monkeys in the habitat                        246
Total alive monkeys in the habitat                         188
Total dead monkeys in the habitat                          308
Total mutant monkeys in the habitat                         81
Total normal monkeys in the habitat                        415
Average weight of all alive monkeys in the habitat       26.43
Average age of all alive monkeys in the habitat          12.51
Average injuries of all alive monkeys in the habitat      0.28
================================================================
```

Your driver program should simply be the following (DO NOT MODIFY):

```
#include "Classes.h"
int main()
{
		Habitat H;
		H.intialization(50);
		H.showGrid();
		H.invasion(300);
		for (int x = 0; x < 10; x++)
		{
			H.ageAll();
			for (int x = 0; x < 52; x++)
			{
				H.feedAll();
			}
		}
		H.showGrid();
		H.getStatistics();
		H.printAll();

	return 0;
}
```

Note: The fight function in this assignment is define as `void fight(GreenMonkey*);` therefore the fight function from your part one will have to be modified in order to work with this definition.

Penalties ranging from 2 marks to 10 marks

- Hard-coded elements (use constants instead)
- No suitable random number generation
- No checks to stop impossible events, such as a dead monkey having a meal
- No commentary in you source code
- Not compiling, runtime error
- Missing functions
- Functions does the incorrect operations

What is to be submitted?

- The .cpp and .h files of your project

THE END