THE UNIVERSITY
OF THE WITWATERSRAND

Lecturer: **Ritesh Ajoodha**

Laboratory 3
Data Structures and Algorithms
Arrays and Linked Lists
11 August , 2015
Due: 18 August, 2015

Question 1 In a popular business simulation video game, Zoo Tycoon, players create a Zoo. This includes creating exhibits to display their favorite animals. The figure below shows an example of how different animals are put into different exhibits. In the game players can put animals in the same exhibit if the animals share similar resources for survival. For example, a zebra and gazelle can be put into the same exhibit since they both live in a savannah biome.

In this lab you are required to create a class called **Animal** with the private fields called *name* and *sound*. The Animal class should have the following definition:

```
1 class Animal{
```

```
  private:
        string name;
        string sound;
  public:
        Animal(const string& n = "", const string& s = "");
        string getName() const;
        string makeSound() const;
};
```

The public methods of the class Animal are: the constructor, which initialises *name* and *sound*; getName(), which returns the name of the animal; and makeSound(), which returns the sound an animal makes. You must overload the operator << so that when you print an object of the class **Animal**, the program outputs: <Name> " says " <Sound>

Input Main:

```
int main() {
        Animal* monkey = new Animal("Frank", "Squeak");
        Animal* tiger = new Animal("Simba", "Chuff");
        Animal* Dog = new Animal("Petals", "Woof");
        cout<< *monkey <<endl;
        cout<< *tiger <<endl;
        cout<< *Dog << endl;
        return 0;
}
```

Program Output:

```
Frank says Squeak
Simba says Chuff
Petals says Woof
```

Question 2 Extend your program from Question 1 by creating a class **Exhibit** which will be used to collect objects of the class Animals into an array. The Exhibit class should have the following definition:

```
class Exhibit{
  public:
        Exhibit(int maxEnt = 10);
        ~Exhibit();
        void add(const Animal& e);
        Animal& operator [](size_t i){ return entries[i];};
        friend ostream& operator<<(ostream& out, const Exhibit& obj);
  private:
        int MaxNumberOfAnimals;
        int CurrentNumberOfAnimals;
        Animal* entries;
};
```

Overload the << operator so that when you output an object of the class Exhibit. You program

will output it in the following way: { <AnimalName1>, <AnimalName2>, <AnimalName3>, ... }

Input Main:

```
int main() {
        Animal* monkey = new Animal("Max", "Eeeeep");
        Animal* tiger = new Animal("Jack", "Roar");
        Animal* Dog = new Animal("Petals", "Woof");
        cout<< *monkey <<endl;
        cout<< *tiger <<endl;
        cout<< *Dog << endl;
        Exhibit cage(10);
        cage.add(*monkey);
        cage.add(*tiger);
        cage.add(Animal("Batty", "Screech"));
        cage.add(Animal("Sheepy", "Bleat␣Bleat"));
        cage.add(Animal("Hippopotamusesy", "growl"));
        cage.add(Animal("Turkey","Gobble"));
        cout << cage << endl;
        cout << cage << endl;
        return 0;
}
```

Program Output:

```
Max says Eeeeep
Jack says Roar
Petals says Woof
{ Max Jack Batty Sheepy Hippopotamusesy Turkey }
{ Max Jack Batty Sheepy Hippopotamusesy Turkey }
```

Question 3 Modify your program so that it deletes the last member of your list with the remove function. The Exhibit class definition should be as follows:

```
class Exhibit{
  public:
        Exhibit(int maxEnt = 10);
        ~Exhibit();
        void add(const Animal& e);
        void removeLast();
        Animal& operator [](size_t i){ return entries[i];};
        friend ostream& operator<<(ostream& out, const Exhibit& obj);
  private:
        int MaxNumberOfAnimals;
        int CurrentNumberOfAnimals;
        Animal* entries;
};
```

Input Main:

```
int main() {
```

```
        Animal* monkey = new Animal("Frank", "Squeak");
        Animal* tiger = new Animal("Simba", "Chuff");
        Animal* Dog = new Animal("Petals", "Woof");
        cout<< *monkey <<endl;
        cout<< *tiger <<endl;
        cout<< *Dog << endl;
        Exhibit cage(10);
        cage.add(*monkey);
        cage.add(*tiger);
        cage.add(Animal("Batty", "Screech"));
        cage.add(Animal("Sheepy", "Bleat␣Bleat"));
        cage.add(Animal("Hippopotamusesy", "growl"));
        cage.add(Animal("Turkey","Gobble"));
        cout << cage << endl;
        cage.removeLast();
        cout << cage << endl;
        cage.removeLast();
        cout << cage << endl;
        return 0;
}
```

Program Output:

```
Frank says Squeak
Simba says Chuff
Petals says Woof
{ Frank Simba Batty Sheepy Hippopotamusesy Turkey }
{ Frank Simba Batty Sheepy Hippopotamusesy }
{ Frank Simba Batty Sheepy }
```

Question 4 In the example of linked lists we did in class, our *C++* code could only add nodes to the front and back of the array. Implement the following new class definition:

```
class StringNode{
  public:
        string elem;
        StringNode* next;
  friend class StringLinkedList;
};

class StringLinkedList{
  public:
        StringLinkedList();
        ~StringLinkedList();
        bool isEmpty() const;
        const string& front() const;
        void addFront(const string& e);
        void removeFront();
        void addBack(const string& s);
```

```
        void removeBack();
        friend ostream& operator<<(ostream& out, const StringLinkedList& obj);
  private:
        StringNode* head;
};
```

Input Main:

```
int main(void){
        StringLinkedList* myList = new StringLinkedList();
        myList->addFront("Massi");
        myList->addFront("Prince");
        cout<< *myList << endl;
        myList->addFront("Conrad");
        myList->addFront("David");
        myList->addFront("Joel");
        cout<< *myList << endl;
        myList->addFront("Ernest");
        myList->addFront("Lindo");
        myList->addFront("Nic");
        cout<< *myList << endl;
        myList->addFront("Sasha");
        myList->removeFront();
        cout<< *myList << endl;
        myList->removeFront();
        cout<< *myList << endl;
        myList->addBack("Jesse");
        myList->addBack("Shane");
        myList->addBack("Richard");
        cout<< *myList << endl;
        myList->removeBack();
        cout<< *myList << endl;
        myList->removeBack();
        cout<< *myList << endl;
        myList->removeBack();
        cout<< *myList << endl;
        myList->removeBack();
        cout<< *myList << endl;
        return 0;
}
```

Program Output:

```
Prince Massi
Joel David Conrad Prince Massi
Nic Lindo Ernest Joel David Conrad Prince Massi
Nic Lindo Ernest Joel David Conrad Prince Massi
Lindo Ernest Joel David Conrad Prince Massi
Lindo Ernest Joel David Conrad Prince Massi Jesse Shane Richard
```

```
Lindo Ernest Joel David Conrad Prince Massi Jesse Shane
Lindo Ernest Joel David Conrad Prince Massi Jesse
Lindo Ernest Joel David Conrad Prince Massi
Lindo Ernest Joel David Conrad Prince
```