ENSF 337 Fall 2018: Tutorial 7

M. Moussavi

Problem I: Consider the following definition and implementation of class called `MyString`, the given main function, and draw a memory diagram for point one and point two when it reaches for the first time and second time.

```
// File: mystring.h
class MyString {
public:
  MyString();
  MyString(const char *s);
  const char* c_str() const;
  int length() const {return lengthM;}
private:
  int lengthM;
  char *storageM;
};
```

```
// File: mystring.cpp
#include "mystring.h
MyString::MyString():lengthM(0), storageM(new char[1]){
  storageM[0] = '\0';
  // POINT ONE
}

MyString::MyString(const char *s): lengthM(strlen(s)){
  storageM = new char[lengthM + 1];
  strcpy(storageM, s);
  // POINT TWO (for first and second time)
}
```

```
#include "mystring.h
int main(void){
  MySTring s;
  MyString s1 ("Blue");
  MyString* s2 = new MyString("Red");
  MyString s3 ("AB");
  return 0;
}
```

Problem II: Consider the partial definition of the following class, and write the implementations of the constructor and other member functions:

```
typedef int Type;  // Type is an alias name for int. You may change int to any type

class SimpleArray {
public:
  SimpleArray(const Type *a, int size);
  // REQUIRES: size > 0, a points to an array of Type
  // PROMISES: dynamically allocates memory and initializes all elements of storageM with values in the array that a
  // pointer a points to.
  int size() const;
  Type get_element(int index) const;
  // REQUIRES: 0 <= index < sizeM
  // PROMISES: returns the value of storageM at index.
  void set_element(int index, Type new_value);
  // REQUIRES: 0 <= index < sizeM
  // PROMISES: sets a new value to storageM at index
private:
  int sizeM;
  Type *storageM;
};
```