Course: Principles of Software Design - ENSF 480

Lab #: 1

Instructor Name: Mahmood Moussavi

Student Name: Samiul Haque, Elias Poitras-Whitecalf

Lab Section: B02

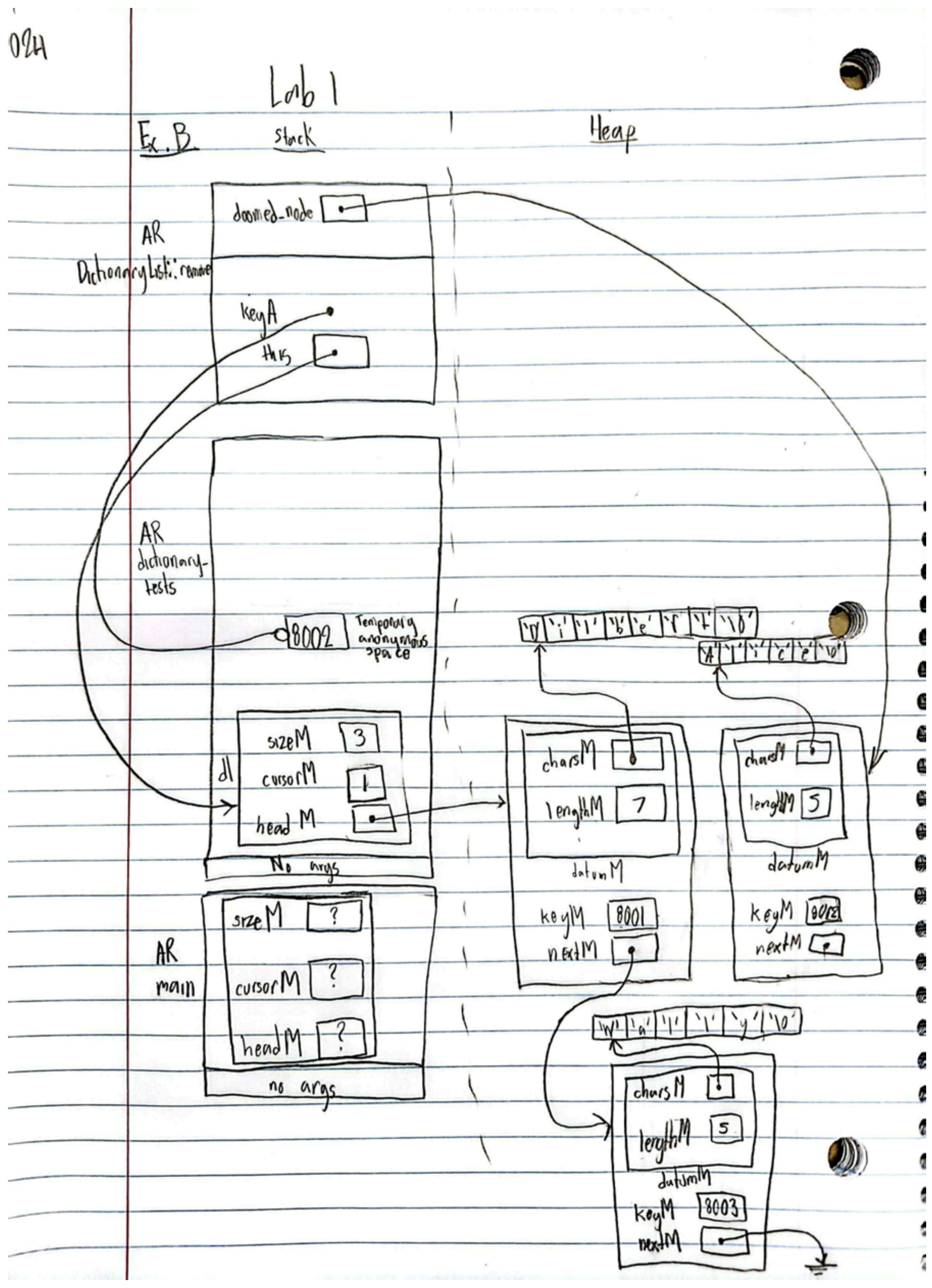
Date Submitted: Sept 13, 2024

Exercise A:

Program output and its order	Your explanation (why and where is the cause for this output)
constructor with int argument is called.	It is called at line 12 in exAmain. The statement, Mystring c = 3 is interpreted by the compiler as a call to the constructor Mystring::Mystring(int n).
default constructor is called. **default constructor is called.**	Both are called at line 18 in exAmain. The statement, Mystring x[2] is interpreted by the compiler as two calls to the default constructor Mystring::Mystring() because two objects are being created.
constructor with char* argument is called.	It is called at line 22 in exAmain. The statement, Mystring* z is interpreted by the compiler as a call to the constructor Mystring::Mystring(const char *s).
copy constructor is called. **copy constructor is called.**	Both are called at line 24 in exAmain. The statement, x[0].append(*z).append(x[1]) is interpreted by the compiler as two calls to the copy constructor Mystring::Mystring(const Mystring& source) because in order to append z and x[1] respectively, a copy of them must be made.
destructor is called. **destructor is called.**	Both are called after line 24 in exAmain. This is because the *z and the x[1] copies are now out of the scope. The variables leaving the scope is interpreted as two called to the destructor.
copy constructor is called.	It is called at line 26 in exAmain. The statement, Mystring mars = x[0] is interpreted by the compiler as a call to the copy constructor Mystring::Mystring(const Mystring& source) because mars is a new object being assignend the value of a preexisting object.
assignment operator called.	It is called at line 28 in exAmain. The statement, x[1] = x[0] is interpreted by the compiler as a call to the assignment operator Mystring& Mystring::operator

	=(const Mystring& S) because both x[0] and x[1] are pre-existing.
constructor with char* argument is called. **constructor with char* argument is called.**	They are called at line 30 and 32 respectively. The statements Mystring.jupiter(“White”) and ar[0] = new Mystring(“Yellow”) are both interpreted by the compiler as calls to the constructor Mystring::Mystring(const char *s).
destructor is called. **destructor is called.** **destructor is called.** **destructor is called.** **destructor is called.**	The destructor, Mystring::~Mystring(), is called 5 times at **line 34**. Four of the five calls to the destructor occur in the cleanup process of the following Mystring objects that leave the scope: x[0], x[1], mars, Jupiter. The fifth call to the destructor is a result of **line 37** delete a[0] which is interpreted by the compiler as a call to the destructor.
constructor with char* argument is called.	It is called at line 39 in exAmain. The statement, Mystring d = “Green” is interpreted by the compiler as a call to the constructor with char* argument Mystring::Mystring(const char *s)
Program terminated successfully.	This is called on line 41 because of the cout statement.
destructor is called. **destructor is called**	The destructor, Mystring::~Mystring(), is called twice at line 43. They both occur in the cleanup process of the Mystring objects c and d, wherein they leave the scope.

Exercise B:


02H
Lab 1
Ex. B
Stack
Heap
doomed_node
AR
DictionaryList::remove
keyA
this
AR
dictionary-tests
8002
Temporary anonymous space
dl
sizeM 3
cursorM 1
head M
No args
AR
main
sizeM ?
cursorM ?
headM ?
no args
'D' 'i' 'l' 'b' 'e' 'r' 't' '\0'
'A' 'l' 'i' 'c' 'e' '\0'
charsM
lengthM 7
datumM
keyM 8001
nextM
charsM
lengthM 5
datumM
keyM 8002
nextM
'W' 'a' 'l' 'l' 'y' '\0'
charsM
lengthM 5
datumM
keyM 8003
nextM

Exercise C:

```
#include <string>
#include <vector>
using namespace std;

struct Company{
    private:
    string companyName;
    Address companyAddress;

    vector <Employee> employees;        //vector of information about
employee's information
                                     //(name, address, date of birth)
    Date dateEstablished;            //the data that company was
established

    vector <Customer> customers;       //vector of information about
customers
                                     //name, address, phone
};

class Date{
    private:
        int day;
        int month;
        int year;
};

class Name{
    private:
        string firstName;
        bool hasMidName;
        string midName;
        string lastName;
};

class Person{
    protected:
        Name name;
        Address address;
        Date dob;
};

class Customer : public Person{
    string phoneNumber;
};

class Employee : public Person{
    Status State;
};

class Status{
    public:
        enum State{active, suspended, retired, fired};

    private:
        State currentState;
};

class Address{
    private:
        int aptNum;
        string streetName;
        string postalCode;
        string city;
        string province;
        string country;
};
```

Exercise D:

human_program.cpp

```
#include <string>
#include <vector>
using namespace std;

struct Company{
    private:
    string companyName;
    Address companyAddress;

    vector <Employee> employees;        //vector of information about
employee's information
                                    //(name, address, date of birth)
    Date dateEstablished;           //the data that company was
established

    vector <Customer> customers;       //vector of information about
customers
                                    //name, address, phone
};

class Date{
    private:
        int day;
        int month;
        int year;
};

class Name{
    private:
        string firstName;
        bool hasMidName;
        string midName;
        string lastName;
};

class Person{
    protected:
```

```cpp
        Name name;
        Address address;
        Date dob;
};

class Customer : public Person{
    string phoneNumber;
};

class Employee : public Person{
    Status State;
};

class Status{
    public:
        enum State{active, suspended, retired, fired};

    private:
        State currentState;
};

class Address{
    private:
        int aptNum;
        string streetName;
        string postalCode;
        string city;
        string province;
        string country;
};
```

human_program.h

```
#ifndef POINT_HUMAN_H
#define POINT_HUMAN_H
#include <cstring>
#include <iostream>
using namespace std;


class Point{
    private:
        double x;     // x coordinate of a location on Cartisian Plain
        double y;     // y coordinate of a location on Cartisian Plain
    public:
        Point(double a = 0, double b = 0); //ctor
        void set_x(double a);    //setter
        void set_y(double a);    //setter
        double get_x() const;    //getter
        double get_y() const;    //getter
        ~Point();
};

class Human {
    protected:
        Point location;   // Location of an object of Human on a Cartisian
Plain
        char *name;       // Human's name
    public:
        Human();    //default ctor
        Human(const char* nam, double x, double y); //ctor with const
char* , double x, doule y args
        ~Human(); //destructor

        char* get_name() const;
        void set_name(const char* name);

        Point get_point() const;
        void set_point(double x, double y);

        void display() const;
};
```

```
#endif
```