**Course: ENSF 614–Fall2021**

**Lab #: Lab 3**

**Student Names: Graydon Hall, Jared Kraus**

**Submission Date: 2021-10-04**

# Exercise A

Diagram

Description automatically generated

Diagram

Description automatically generated

# Exercise B

# Diagram Description automatically generated

# Diagram, engineering drawing Description automatically generated

Diagram

Description automatically generated Diagram

Description automatically generated

# Exercise C

|  |
| --- |
| */\**  *\* File Name: lab3Clock.h*  *\* Lab # and Assignment #: Lab 3 Exercise C*  *\* Lab section: 1*  *\* Completed by: Graydon Hall and Jared Kraus*  *\* Submission Date: 2021-10-04*  *\*/*  #ifndef lab3\_exe\_C\_Clock  #define lab3\_exe\_C\_Clock  */\* This class definition represents a clock that shows hours,*  *\* Minutes, and Seconds.*  *\*/*  class Clock{      public:          Clock();  *// PROMISES: initializes the values of hours, minutes, and seconds, all to 0*          Clock(int seconds);  *//PROMISES: receives integer argument in seconds, uses it to set clock time in*  *// hours, minutes, and seconds*          Clock(int hours, int minutes, int seconds);  *// PROMIESES: initializes hours, minutes, and seconds for clock based on user provided arguments*          int get\_hour () const;  *// PROMISES: return hour value for clock*          int get\_minute () const;  *// PROMISES: return minute value for clock*          int get\_second () const;  *// PROMISES: return second value for clock*          void set\_hour (int arg);  *// PROMISES: set a new value for hour with the value of arg*          void set\_minute (int arg);  *// PROMISES: set a new value for minute with the value of arg*          void set\_second (int arg);  *// PROMISES: set a new value for second with the value of arg*          void increment ();  *// PROMISES: increments the value of the clock’s time by one second.*          void decrement ();  *// PROMISES: decrements the value of the clock’s time by one second.*          void add\_seconds (int second);  *// PROMISES: adds the value of given seconds to the value of the current time*  *// REQUIRES: a positive integer argument in seconds*        private:          int hour;*// hours on the clock*          int minute;*// minutes on the clock*          int second;*// seconds on the clock*          int hms\_to\_sec();  *// PROMISES: returns the total value of data members in a Clock object,*  *// in seconds.*          void sec\_to\_hms(int n);  *// PROMISES:  receives an argument n in seconds, and sets the values*  *// for the Clock data members, second, minute, and hour, based on this argument.*      };  #endif |

|  |
| --- |
| */\**  *\* File Name: lab3Clock.cpp*  *\* Lab # and Assignment #: Lab 3 Exercise C*  *\* Lab section: 1*  *\* Completed by: Graydon Hall and Jared Kraus*  *\* Submission Date: 2021-10-04*  *\*/*  #include <iostream>  #include <iomanip>  using std::cout;  using std::endl;  using std::setw;  using std::setfill;  #include "lab3Clock.h"  void print(const Clock& t);  *// Start GH Code*  Clock::Clock(){      hour=0;      minute=0;      second=0;  }  Clock::Clock(int seconds)  {  *// set values for hour minute second based on seconds argument*      sec\_to\_hms(seconds);  }  Clock::Clock(int hours, int minutes, int seconds)  {      if(hours < 0 || hours > 23 || minutes < 0      || minutes > 59 || seconds < 0 || seconds > 59){          hour=0;          minute=0;          second=0;      }      else {          hour = hours;          minute = minutes;          second = seconds;      }  }  int Clock::get\_hour () const {      return hour;  }    int Clock::get\_minute () const {      return minute;  }  int Clock::get\_second () const {      return second;  }  void Clock::set\_hour (int arg){      if(arg>=0 && arg <=23){          hour=arg;      }  }  void Clock::set\_minute (int arg){      if(arg>=0 && arg <=59){          minute=arg;      }  }  void Clock::set\_second (int arg){      if(arg>=0 && arg <=59){          second=arg;      }  }  void Clock::increment (){      if(second==59){          if(minute==59){              if(hour==23){                  hour=0;                  minute=0;                  second=0;              }else{                  hour++;                  minute=0;                  second=0;              }          }else{              minute++;              second=0;          }      } else{          second++;      }  }  void Clock::decrement (){      if(second==0){          if(minute==0){              if(hour==0){                  hour=23;                  minute=59;                  second=59;              }else{                  hour--;                  minute=59;                  second=59;              }          }else{              minute--;              second=59;          }      } else{          second--;      }  }  *// maybe find more efficient way to do this later*  void Clock::add\_seconds(int seconds){      if(seconds>0){          for (int i = 0; i < seconds; i++) {              increment();          }      }  }  int Clock::hms\_to\_sec(){      int total\_seconds;      total\_seconds = hour\*3600+minute\*60+second;  }  void Clock::sec\_to\_hms(int n){      if(n<0){          n=0;      }      int full\_hours = n/3600;      int remaining\_secs = n - full\_hours\*3600;      while(full\_hours>23){          full\_hours = full\_hours - 24;      }      int full\_minutes = remaining\_secs/60;      remaining\_secs = remaining\_secs - full\_minutes\*60;        hour = full\_hours;      minute = full\_minutes;      second = remaining\_secs;  } |

# Exercise D – Part 1 – Step 1

Diagram, box and whisker chart

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

# Exercise D – Part 1 – Step 2

**1. At point four in the main function, how many times the constructor of class DynString is called?**

By the time we get to point 4, the main constructor will have been called 2 times. One time each for the variables a, and b.

**2. At point four how many times the destructor of the class DynString is called?**

By the time we get to point 4, the destructor will have been called twice. First it is called for c, since the variable c ceases to exist outside of the curly braces it was created in. The second time the destructor is called is during is when the “delete b” line of code is executed.

**3. Overall how many times in total the destructor of the class DynString will be called in this program?**

By the time we get to the end of the program, the destructor of DynString is called 3 times, once for a, b, and c each.

**4. Answer the question that is noted in the main function: What is going wrong after you press the return key in the main function?**

* In this program, the variable c is set equal to the variable a within an extra pair of curly braces, so the destructor is called for c after the program exits these extra curly braces (since c goes out of scope).
* In the destructor for the DynString class, the allocated space for storageM is freed through the line delete [] storageM; but the pointer pointing to that spot is still pointing there.
* Since c was set equal to a, their storageM pointer pointed to the same location in heap memory.
* Therefore, when the destructor for a is called again at the end of the main function, and we try to de-allocate the memory for storageM that was already de-allocated, we get an error.
* A way to solve this would be adding storageM = nullptr within the destructor for the DynString class.

# Exercise D – Part 2

Here is a copy of our append function

|  |
| --- |
| void DynString::append(const DynString& tail){    int new\_length = lengthM + tail.lengthM;*// find new length*    char \* holder;*// char array to hold our new string*    holder = new char[new\_length];*// add 1 to length to account for '\0'*    strcpy(holder, storageM);*// copy starageM contents into holder first*  *// copy contents of tail into holder*    for (int i = 0; i < tail.lengthM; i++) {      holder[i+lengthM] = tail.storageM[i];    }    holder[new\_length] = '\0';    storageM = holder;    lengthM = new\_length;  } |

Program output:

Text

Description automatically generated