ENSF 614 – Fall 2021

Lab 3 – Tuesday, October 05

Student Name: Bhavyai Gupta

Submission date: October 05, 2021

# Exercise A – AR Diagram for Point One

Diagram

Description automatically generated

# Exercise A – AR Diagram for Point Two

Diagram

Description automatically generated

# Exercise B – AR Diagram for Point One

Diagram

Description automatically generated

# Exercise B – AR Diagram for Point Two

Diagram

Description automatically generated

# Exercise B – AR Diagram for Point Three

Diagram

Description automatically generated

# Exercise B – AR Diagram for Point Four

Diagram

Description automatically generated

# Exercise C – Source File lab3Clock.h

/\*

 \* File Name:               lab3Clock.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 3 Exercise C

 \* Lab section:             B01

 \* Completed by:            Bhavyai Gupta

 \* Submission Date:         October 05, 2021

 \*/

#ifndef LAB\_3\_CLOCK

#define LAB\_3\_CLOCK

class Clock

{

private:

    int hour;

    int minute;

    int second;

    int hms\_to\_sec();

    void sec\_to\_hms(int n);

public:

    Clock();

    // Constructs an object of Clock with all values initialized to 0

    //

    // REQUIRES

    //      NA

    //

    // PROMISES

    //      An object of Clock with all values initialized to 0

    Clock(int s);

    // Constructs an object of Clock with time equivalent to the value

    // of s passed in the argument

    //

    // REQUIRES

    //      s >= 0

    //

    // PROMISES

    //      An object of Clock with time equivalent to the value of s

    Clock(int h, int m, int s);

    // Constructs an object of Clock with parameters h, m, s

    //

    // REQUIRES

    //      h >=0 and h <= 23

    //      m >= 0 and m <= 59

    //      s >= 0 and s <= 59

    //

    // PROMISES

    //     an object of Clock with time h, m, and s

    int get\_hour() const;

    // Getter for data member hour

    //

    // REQUIRES

    //      NA

    //

    // PROMISES

    //      Returns the hour of the Clock

    int get\_minute() const;

    // Getter for data member minute

    //

    // REQUIRES

    //      NA

    //

    // PROMISES

    //      Returns the minute of the Clock

    int get\_second() const;

    // Getter for data member second

    //

    // REQUIRES

    //      NA

    //

    // PROMISES

    //      Returns the second of the Clock

    void set\_hour(int h);

    // Setter for data member hour

    //

    // REQUIRES

    //      h >= 0 and h <= 23

    //

    // PROMISES

    //     Sets the hour with the value passed in parameter

    void set\_minute(int m);

    // Setter for data member minute

    //

    // REQUIRES

    //      m >= 0 and m <= 59

    //

    // PROMISES

    //     Sets the minute with the value passed in parameter

    void set\_second(int s);

    // Setter for data member second

    //

    // REQUIRES

    //      s >= 0 and s <= 59

    //

    // PROMISES

    //     Sets the second with the value passed in parameter

    void increment();

    // Increment the time by one second

    //

    // REQUIRES

    //      NA

    //

    // PROMISES

    //      Increases the time represented in the object by one second

    void decrement();

    // Decrement the time by one second

    //

    // REQUIRES

    //      NA

    //

    // PROMISES

    //      Decreases the time represented in the object by one second

    void add\_seconds(int s);

    // Add s seconds to the time represented by the object

    //

    // REQUIRES

    //      s >= 0

    //

    // PROMISES

    //      Time represented in increased by s seconds

};

#endif

# Exercise C – Source File lab3Clock.cpp

/\*

 \* File Name:               lab3Clock.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 3 Exercise C

 \* Lab section:             B01

 \* Completed by:            Bhavyai Gupta

 \* Submission Date:         October 05, 2021

 \*/

#include "lab3Clock.h"

Clock::Clock()

{

    set\_hour(0);

    set\_minute(0);

    set\_second(0);

}

Clock::Clock(int s)

{

    if (s < 0)

    {

        set\_hour(0);

        set\_minute(0);

        set\_second(0);

    }

    else

    {

        sec\_to\_hms(s);

    }

}

Clock::Clock(int h, int m, int s)

{

    if (s >= 0 && s <= 59)

    {

        if (m >= 0 && m <= 59)

        {

            if (h >= 0 && h <= 23)

            {

                set\_hour(h);

                set\_minute(m);

                set\_second(s);

                return;

            }

        }

    }

    set\_hour(0);

    set\_minute(0);

    set\_second(0);

}

void Clock::sec\_to\_hms(int n)

{

    int h = (n / 3600);

    int m = (n / 60) - (h \* 60);

    int s = n - (m \* 60) - (h \* 3600);

    h = h % 24;

    set\_hour(h);

    set\_minute(m);

    set\_second(s);

}

int Clock::hms\_to\_sec()

{

    return (get\_hour() \* 3600) + (get\_minute() \* 60) + get\_second();

}

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 h)

{

    if (h >= 0 && h <= 23)

    {

        hour = h;

    }

}

void Clock::set\_minute(int m)

{

    if (m >= 0 && m <= 59)

    {

        minute = m;

    }

}

void Clock::set\_second(int s)

{

    if (s >= 0 && s <= 59)

    {

        second = s;

    }

}

void Clock::increment()

{

    if (second < 59)

    {

        second++;

    }

    else

    {

        second = 0;

        if (minute < 59)

        {

            minute++;

        }

        else

        {

            minute = 0;

            if (hour < 23)

            {

                hour++;

            }

            else

            {

                hour = 0;

            }

        }

    }

}

void Clock::decrement()

{

    if (second > 0)

    {

        second--;

    }

    else

    {

        second = 59;

        if (minute > 0)

        {

            minute--;

        }

        else

        {

            minute = 59;

            if (hour > 0)

            {

                hour--;

            }

            else

            {

                hour = 23;

            }

        }

    }

}

void Clock::add\_seconds(int s)

{

    for (int i = 0; i < s; i++)

    {

        increment();

    }

}

# Exercise C – Program Output

Text

Description automatically generated

# Exercise D – AR Diagram for Point One – First Time

Diagram

Description automatically generated

# Exercise D – AR Diagram for Point One – Second Time

Diagram

Description automatically generated

# Exercise D – AR Diagram for Point Three

Diagram, engineering drawing

Description automatically generated

# Exercise D – AR Diagram for Point Four

Diagram, engineering drawing

Description automatically generated

# Exercise D – Answers to questions

1. Till point 4, the constructor of DynString has been called twice.
2. Till point 4, the destructor of DynString has been called twice.
3. Destructor of DynString would be called thrice during the program execution.
4. Object “c” was a shallow copy of object “a”. Hence, the storageM pointer in object “c” pointed to the same location as that of storageM pointer in object “a”.

When the object “c” goes out of scope during the program, its destructor was called which also deallocates the memory pointed by storageM pointer. This means, the storageM pointer of object “a” is now pointing to de-allocated area on the heap.

Just before the program ends, object “a” goes out of scope and its destructor is called. Since memory pointed by storageM is already deleted, when its tried to be deleted again in the destructor, we run into error and the program exits with non-zero return code.

# Exercise D – Source Code of append

void DynString::append(const DynString &tail)

{

    // allocate a new array of the right length

    char \*temp = new char[lengthM + tail.lengthM + 1];

    // copy whatever characters need to be copied into the new array

    for(int i=0; i<lengthM; i++)

    {

        temp[i] = storageM[i];

    }

    for(int i=0; i<tail.lengthM; i++)

    {

        temp[i + lengthM] = tail.storageM[i];

    }

    temp[lengthM + tail.lengthM] = '\0';

    // deallocate the old array

    delete[] storageM;

    storageM = temp;

    // adjust the value of the lengthM variable

    lengthM = lengthM + tail.lengthM;

}

# Exercise D – Program Output

Text

Description automatically generated