UEE1303(1031) S21
Midterm Examination

FULL SCORES:
120 %

EXAMINATION TIME:
18：30～21：30, total 180 minutes

INSTRUCTIONS:

This midterm examination is composed of 5 different problem sets. **Your total score will be the summation of the scores from the highest 3 problem sets you've solved. If you only finish problem A, B, C, the highest score you can get is 100 points.** You are allowed to open any notes or books, but prohibited to browse on the internet to search C/C++ codes as direct answers. Read carefully the statements and requirements of each problem. Once you complete your program for one problem, please raise your hand and TA will come to you and test your program. Please note that points will be given until your program fully fulfills the requirements of each problem. No partial credits will be given.

You should be aware of that some lengthy problems are not as difficult as you think. In contrast, some short problems may not be solved easily. Last but not least, **cheating** is a serious academic demeanor and should be avoided at all most. Once any cheating is caught, all your answers will be void and get 0 point for your midterm.

Good luck!

UEE1303(1048) S21: Midterm Examination Demo Sheet

Student ID # : _____,Name :_____

Full Scores: up to 120 points

→You may pick arbitrary numbers of problems to solve. **The total score you obtain will be the summation of the scores from the highest 3 problem sets you've solved. If you only finish problem A, B, C, the highest score you can get is 100 points.**

Problem Set A (40%)		Problem Set B (40%)	
Subproblem	**TA Signature**	**Subproblem**	**TA Signature**
Q1 (10%)		Q1 (15%)	
Q2 (15%)		Q2 (10%)	
Q3 (15%)		Q3 (15%)	
Problem Set C (40%)		**Problem Set D (60%)**	
Subproblem	**TA Signature**	**Subproblem**	**TA Signature**
Q1 (10%)		Q1 (20%)	
Q2 (15%)		Q2 (20%)	
Q3 (15%)		Q3 (20%)	
Problem Set E (60%)			
Subproblem	**TA Signature**		
Q1 (5%)			
Q2 (35%)			
Q3 (20%)			

Total Score: _____ / 120

【Source Code and Testcase】

The path of the source code and the open testcase for each problem set:

```
<PATH> /home/share/midterm/pa ~ /home/share/midterm/pe
```

Copy the all the source codes and open testcases to your local directory and start to complete each problem in the corresponding folder.

```
<EX> $ cp -R /home/share/midterm/ ~/.
```

【PROBLEM SET A】(40%)

Three-dimensional vector in float is a common datatype used in Computer Graphics packages, like Unity. Given the definition of class **CFlt3** in **pg0A.h** and the main program in **main.cpp**, complete the tasks in each problem. You can modify the file **pg0A.cpp** and add proper prototypes in **pg0A.h**.

```
//pg0A.h
class CFlt3 {
private:
  float x;
  float y;
  float z;

public:
  //Q1 Basics
  CFlt3();                        //x=y=z=0.0
  CFlt3(float a);                //x=y=z=a
  CFlt3(float a, float b, float c);   //x=a,y=b,z=c
  CFlt3(const CFlt3& t);          //x=t.x,y=t.y,z=t.z
  void setXYZ(float a, float b, float c); //x=a,y=b,z=c
  float getX();                  //return x
  float getY();                  //return y
  float getZ();                  //return z
  friend ostream& operator<<(ostream& out, const CFlt3& t);

  //Q2 overload add(+), subtract(-) and equality(==) for CFlt3
  //2-1: overload operator + for addition
  //2-2: overload operator – for subtraction
  //2-3: overload operator == for equality; return true if
respective private data elements are all equal.

  //Q3: inner product (*), cross product (^) and distance (dist)
  //3-1: overload operator * for inner product
  //ex: a*b=a.x*b.x+a.y*b.y+a.z*b.z
  //3-2: overload operator ^ for cross product
  //ex: a^b=(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x)
  //3-3: implement a friend function distance
  //ex: dist(a,b)=sqrt((a.x-b.x)^2+(a.y-b.y)^2+(a.z-b.z)^2)
};
```

Note that you need to pass both the test of t1 and t2 to get the points of each questions (Q1, Q2 and Q3).

A-01 BASICS (10 points)

Please complete all constructors and the required member functions in each above class definition with the following requirements. Given test Q1 in the main function for this problem,

```
#include "pg0A.h"
int main(){
  CFlt3 v0;
  CFlt3 v1(1.0);
  const CFlt3 v2(2.0,2.0,2.0);
  const CFlt3 v3(v2);
  cout << fixed << showpoint << setprecision(2);
#ifdef Q1
  cout << v0.getX() << endl;
  cout << v1.getY() << endl;
  cout << v2.getZ() << endl;
  cout << v3 << endl;
#endif
}
```

(1) There are four constructors to create **CFlt3** objects: **CFlt3(), CFlt3(float), CFlt3(float, float, float)** and **CFlt3(const CVect3 &)**.

(2) **setXYZ(float, float, float)** , a mutator function, is used to set values to private data memebers whereas **getX()**, **getY()** and **getZ()** are three accessor functions to read private data members.

(3) **operator<<** is a given friend member function that overloads the operator **<<** to output the private data member as **(x,y,z)** to ostream **out**.

The sample output for the Q1 part of t1 and t2:

```
>make t1q1
>./t1
0.00
1.00
2.00
(2.00,2.00,2.00)
>
```

```
>make t2q1
>./t2
0.00
-1.00
1.00
(-1.00,-1.00,-1.00)
>
```

A-02 (15 points)

Give test Q2 in the main function for this problem,

```
#ifdef Q2
     const CFlt3& v4 = v1+v2+3.0; cout << v4 << endl;
     CFlt3* pv5 = new CFlt3(v4-v1); cout << (*pv5) << endl;
     if ((*pv5) == v2)
     cout << "pv5 is equal to v2." << endl;
    else
     cout << "pv5 is not equal to v2." << endl;
#endif
```

Please complete three overloading operators (`+`, `-` and `==`) to show the corresponding information on screen.

(1) Since **`(1.0+1.0+3.0,1.0+2.0+3.0,1.0+2.0+3.0)=(6.0,6.0,6.0),`** please show on screen **`(6.0,6.0,6.0)`** after calling **`cout << v4 << endl;`**

(2) Since **`(6.0-1.0,6.0-1.0,6.0-1.0)=(5.0,5.0,5.0),`** please show on screen **`(5.0,5.0,5.0)`** after calling **`cout << (*pv5) << endl;`**

(3) Since contents of objects that **`pb5`** points to and of **`v2`** are not equal, return false to show on screen **`"pv5 is not equal to v2."`**

The sample output for the Q2 part of t1 and t2:

```
>make t1q2
>./t1
(6.00,6.00,6.00)
(5.00,5.00,5.00)
pv5 is not equal to v2.
>
```

```
>make t2q2
>./t2
(2.50,1.50,0.50)
(2.50,1.50,0.50)
pv5 is equal to v4.
>
```

A-03 (15 points)

Given test Q3 in the main function for this problem:

```
#ifdef Q3
    const CFlt3 v6 = CFlt3(v1*v3); cout << v6 << endl;
v0.setXYZ(0.5,-1.0,-0.5);
    CFlt3 v7 = v6^v0; cout << v7 << endl;
    cout << "distance(v6, v7)=" << dist(v6, v7) << endl;
#endif
```

(1) Implement the overloading operator `*` for the inner product of two **`CFlt3`** objects; For example, `a*b=a.x*b.x+a.y*b.y+a.z*b.z`

(2) Implement the overloading operator `*` for the cross product of two **`CFlt3`** objects; For example, `a^b=(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x)`

(3) Implement a friend function to compute the distance of two **`CFlt3`** objects; That is, `dist(a,b)=`$\sqrt{(a.x-b.x)^2+(a.y-b.y)^2+(a.z-b.z)^2}$

The sample output

```
>make t1q3
>./t1
(6.00,6.00,6.00)
(3.00,6.00,-9.00)
distance(v6, v7)=15.30
>
```

```
>make t2q3
>./t2
(3.00,3.00,3.00)
(3.70,-4.80,-1.50)
distance(v6, v7)=9.03
>
```

【Problem Set B】(40%)

In this problem, you should implement two account classes, one is **`TWDAccount`**, the other is **`USDAccount`**.

The function and variable of classes you should add are illustrated in the main function.

Class **`TWDAccount`** is described in **`TWDAccount.h`**.
Class **`USDAccount`** is described in **`USDAccount.h`**.
Main function is described in **`mid{1-3}.cpp.`**
These files are puts in ***/home/share/midterm/pb/.***

Most important, you are not allowed to modify the function or variable we give, but you can add the property, for example:

```
int a -> static int a // add property is allowed
int a -> double a //not allowed
int a -> int c //not allowed
```

There are three parts (15%, 10%, 15%) in this problem.

<Compile>
B-01(15%):

```
g++ TWDAccount.cpp USDAccount.cpp mid1.cpp -o ./prob_b
```

B-02(10%):

```
g++ TWDAccount.cpp USDAccount.cpp mid2.cpp -o ./prob_b
```

B-03(15%):

```
g++ TWDAccount.cpp USDAccount.cpp mid3.cpp -o ./prob_b
```

The content of **`TWDAccount.h`** as shown in below:

```
class TWDAccount{
    int amount;
    //the number of TWDAccount user
     static int total_user_number;
     string user_name;
    public:
          void change_user_info(string new_user_name)const{
user_name = new_user_name;};
};
```

The content of **`USDAccount.h`** as shown in below:

```
class USDAccount{
     double amount;
    public:
          static const double EXCHANGE_RATE;
};
```

The content of **mid3.cpp**(which includes **mid1.cpp & mid2.cpp**) as shown in below:

```
#include <iostream>
#include <string>
#include "TWDAccount.h"
#include "USDAccount.h"
using namespace std;

const double USDAccount::EXCHANGE_RATE = 30.0;

int main(){
    ///
    ///PART1(15%): implement constructor, friend function, operator
    ///
    TWDAccount *A = new TWDAccount(165, "tom");
    USDAccount *B = new USDAccount(180);
    const TWDAccount *A_const = A;

    //use friend function to get variable USDAccount::amount
    cout << "USD dollar: " << get_amount_by_friend(*B) << endl;

    USDAccount *C = new USDAccount(180);
    USDAccount *D = new USDAccount(180);
    //implement + operator, the amount of (A/30 + B) = the amount of C
    (*C) = (*B) + (*A);
    (*D) = (*C) + (*A);
    cout << "USD dollar: " << get_amount_by_friend(*C) << endl;
    cout << "USD dollar: " << get_amount_by_friend(*D) << endl;

    //implement ++ operator, amount+=1
    (*D)++;
    cout << "USD dollar: " << get_amount_by_friend(*D) << endl;

    ///
    ///PART2(10%): implement operator, type casting
    ///

    //implement * operator, amount*=1.1
    USDAccount E = (*D)*1.1;
    cout << "USD dollar: " << get_amount_by_friend(E) << endl;

    //implement class type casting, int(*A) should return the amount
    USDAccount *T = new USDAccount(int(*A)/USDAccount::EXCHANGE_RATE);
    cout << "USD dollar: " << get_amount_by_friend(*T) << endl;
    //implement the << operator, format is: “out << user_name << " has
" << amount << " dollars";”
    cout << *A << endl;
```

```
    TWDAccount *L = new TWDAccount(210, "cat");
    //implement class type casting, int(*L) should return the amount.
   USDAccount *T2 = new USDAccount(int(*L)/USDAccount::EXCHANGE_RATE);
    cout << "USD dollar: " << get_amount_by_friend(*T2) << endl;
    //implement the << operator
    cout << *L << endl;

    //implement class type casting, if amount<=0, return True,
otherwise, return False
    cout << ((bool)(*L)?"you have no money":"you are rich man") <<
endl;

    //implement the USDAccount constructor with TWDAccount class(*L),
the value of amount should be divided by 30
    USDAccount U = USDAccount(*L);
    cout << "TWD account has " << L->get_amount() << \
     " dollars which can be transferred to USD account , amount is :"\
     << get_amount_by_friend(U) << endl;

    ///
    ///PART3(15%):
    ///

    //change user info
    A_const->change_user_info("bob");
    cout << *A << endl;

     //use “=” to initialize K is not allowed, compiler should report
     //error.
    //USDAccount K = 50;

    //get_total_user_number is a static function,
    //should return total_user_number
    cout << "now total user is:" << \
              TWDAccount::get_total_user_number() << endl;
    delete A;
    cout << "now total user is:" << \
              TWDAccount::get_total_user_number() << endl;
    delete L;
    cout << "now total user is:" << \
              TWDAccount::get_total_user_number() << endl;
    delete B;
    delete T;

    delete C;
    delete D;
    delete T2;
```

```
}
```

In addition, **mid3_error_line70.cpp is used to ensure initialize K is not allowed (line 70 in mid3_error_line70.cpp).**

OUTPUT:

```
USD dollar: 180
USD dollar: 185.5
USD dollar: 191
USD dollar: 192
USD dollar: 211.2
USD dollar: 5.5
tom has 165 dollars
USD dollar: 7
cat has 210 dollars
you are rich man
TWD account has 210 dollars which can be transferred to USD
account , amount is :7
bob has 165 dollars
now total user is:2
now total user is:1
now total user is:0
```

【Problem Set C】(40%)

In mathematics, **the algebra of sets**, defines the properties and laws of sets, the set operations of union (A ∪ B), intersection (A ∩ B), Relative complement (A - B), and the relations of set equality (A = B) and set inclusion (A ⊆ B). It also provides systematic procedures for evaluating expressions, and performing calculations, involving these operations and relations. In this program, you are asked to **implement these set operations by overloading operators**. The definition of operations are given as follows.

1. *Union* **(A ∪ B):** the set of elements which are in *A*, in *B*, or in both *A* and *B*.
2. *Intersection* **(A ∩ B):** the set containing all elements of *A* that also belong to *B*.
3. *Relative complement* **(A − B):** the set of elements in A but not in B.
4. *Equality* **(A = B):** the set A is equals to the set B.
5. *Inclusion* **(A ⊆ B):** every element of set *A* is also in *B*.

The source code (**stu_main.cpp, stu_set.cpp, set.h**) and input file (**input.in**) will be given. You only need to complete **stu_set.cpp** and add some proper codes to **stu_main.cpp**. These codes are located in ***$ /home/share/midterm/pc/**** and you can copy them to your workstation.

<Compile> $ **g++ stu_main.cpp stu_set.cpp -o main**
<Execute> $**./main < input.in**

The input file contains four lines, each line is an individual set including a mount of elements (1 to 10) following by stop point -1. Note that every element in the set is sorted. More specifically, the following code can be expressed as
set1=**[10, 12, 13, 15, 17, 18, 20]**,
set2=**[8, 10, 12, 13, 21, 28],**
set3=**[10, 12, 13, 15, 17, 18, 20]**
set4=**[15, 17, 18]**

Input file: **input.in**

```
10 12 13 15 17 18 20 -1
8 10 12 13 21 28 -1
10 12 13 15 17 18 20 -1
15 17 18 -1
```

Header file**: set.h**

```
#ifndef SET_H
#define SET_H
#include <iostream>
class Set {
private:
    int *data;
    int SIZE;
public:
    //Q-1
    Set();
    Set(int *input_data, const int input_size);
    ~Set();
    void display();

    //Q-2
    friend bool operator==(const Set&, const Set&);//equality
    friend bool operator<(const Set&, const Set&);// inclusion
    friend bool operator>(const Set&, const Set&);// inclusion

    //Q-3
    friend Set operator&&(const Set&, const Set&);// Intersection
    friend Set operator||(const Set&, const Set&);// Union
    friend Set operator-(const Set&, const Set&);// relative complement

};

#endif
```

Q1 BASIC (10%)

In this part. First you should complete the function that can enter the set from the input file. The command line would be $ **./main < input.in**. Next on, complete all constructors, destructors and a member function (display) which have been declared in the header file. The example output is based on the input file that was given in the previous page.

Example: **main_basic.cpp**

```
//main.cpp
#include "set.h"
#include <iostream>
using namespace std;

int main(int argc, char** argv) {

    int Arr[10] = {0};
    int ArrSIZE = 0;
```

```
    //Q-1
    Set set1(Arr, ArrSIZE);
    Set set2(Arr, ArrSIZE);
    Set set3(Arr, ArrSIZE);
    Set set4(Arr, ArrSIZE);

    cout << "set1: "; set1.display();
    cout << "set2: "; set2.display();
    cout << "set3: "; set3.display();
    cout << "set4: "; set4.display();

    return 0;
}
```

Example output

```
set1: 10 12 13 15 17 18 20
set2: 8 10 12 13 21 28
set3: 10 12 13 15 17 18 20
set4: 15 17 18
```

Q2 MODERATE(15%)

In this part, the set operator equality and inclusion will be implemented by yourself. The definition of the overloading operator that has been redefined and corresponds to the set operator is given as follows.

overloading operation	set operation
A == B	*Equality (A = B)*
A < B	*Inclusion (A ⊆ B)*
A > B	*Inclusion (A ⊇ B)*

Example: `main_moderate.cpp` (following main_basic.cpp)

```
int main(int argc, char** argv) {

  //Q-2
  cout << "answer1: " << ((set1 == set2)?"True":"False") << endl;
  cout << "answer2: " << ((set1 == set3)?"True":"False") << endl;
  cout << "answer3: " << ((set4 < set3)?"True":"False") << endl;
  cout << "answer4: " << ((set1 > set3)?"True":"False") << endl;

    return 0;
```

```
}
```

Example output

```
answer1: False
answer2: True
answer3: True
answer4: True
```

<u>Q3 ADVANCED</u>(15%)

Following the Q2, you are asked to implement three different kinds of overloading operator, union, intersection, relative complement. Also, we can combine these three operators to perform more difficult calculations. BE CAREFUL!! you have to do some operation in the operator function body to make sure that the output result is a sorted set.

overloading operation	set operation
A && B	*Union (A ∪ B)*
A || B	*Intersection (A ∩ B)*
A - B	*Relative complement (A - B)*

Example: main_advance.cpp (following main_moderate.cpp)

```
int main(int argc, char** argv) {

    //Q-3
    Set answer5 = (set1 && set2) || (set3 - set2);
    cout << "answer5: ";
    answer5.display();

    Set answer6 = (set1 && set2 && set3) - (set4 || set4);
    cout << "answer6: ";
    answer6.display();

    return 0;
}
```

Example output

```
answer5: 10 12 13 15 17 18 20
answer6: 10 12 13
```

【PROBLEM SET D】(60%)

The portable pixmap format (PPM) is a kind of RGB image file format, which contains a header recording in the first three lines in ASCII.

The first line is a two-byte word that identifies the encoding of this file (ASCII or binary). The word is a capital P followed by a single-digit number. "P3" means this file is encoding in ASCII, and “P6” means that it is encoding in binary. The second line records the width and height of the image in pixels, and the third line is the maximum value of pixels.

Starting from the fourth line, it records the pixels of that image in binary or ASCII. Every three numbers stands for a pixel, representing for R, G, and B channels.

Here are examples of the PPM file:

```
P3
4 4
255
159 184 192 160 185 192 160 185 193 160 185 192
161 185 193 160 185 192 160 184 193 161 186 193
159 185 194 158 185 194 160 186 195 161 186 194
160 185 193 160 185 192 161 185 194 161 186 193
```

The above is the format of a PPM file with pixels encoding in ASCII, and the image size is 4x4 pixels. The pixel value for each color channels would be in the range from 0 to 255.

```
P6
10 10
255
□□□□□□□□□□□□□□□□...
```

The above is the format of a PPM file with pixels encoding in binary, and the image size is 10x10 pixels. The pixel value for each color channels would be in the range from 0 to 255.

Given the definition of class **`PPM`** in **`pg0D.h`** and the main program in **`main.cpp`**, please complete the tasks in each problem. You should complete the file in **`pg0D.cpp`** and add proper prototypes in **`pg0D.h`**.

```
//pg0D.h
#ifndef PG0D_H
#define PG0D_H
#include<fstream>
using std::ifstream;
using std::ofstream;

typedef struct{
    unsigned char R;
    unsigned char G;
    unsigned char B;
}Pixel;

class ppm{
    private:
        int h;               //the height of the picture
        int w;               //the width of the picture
        int pixel_range;    //pixel max value
        Pixel **picture;    //record the RGB layer of the picture

    public:
        //Q1
        ppm();
        ppm(const char*);
        ~ppm();
        bool saveToFile(const char*, const char);
        //Q2
        void operator=(const ppm &);
        ppm crop(int, int, int, int);
        //Q3
        ppm rotate();
        ppm resize(int, int);
};

#endif
```

D-01 **BASICS** (20 points)

Please complete all constructors and the required member functions in each above class definition with the following requirements. Given test Q1 in the main function for this problem,

```
#include <iostream>
#include "pg0D.h"
using namespace std;

int main(){
    //Q1(20%)
    cout<<"Q1"<<endl;
    ppm pic1("t1.ppm");
        cout<<(pic1.saveToFile("t1_1.ppm",  'a')?"write  a  file
t1_1.ppm ":"fail to write a ASCII file")<<endl;
       cout<<(pic1.saveToFile("t1_2.ppm",  'b')?"write  a  file
t1_2.ppm ":"fail to write a binary file")<<endl;
    ...
}
```

(4) In `PPM` object, there are two constructor functions and one destructor functions: **`PPM(), PPM(`**`const char*`**`)`** and **`~PPM()`**. **`PPM()`** and **`~PPM()`** is the default constructor and destructor you need define by yourself. **`PPM(`**`const char*`**`)`** would read the specified image and initialize the object with that image.

(5) `saveToFile(const char* f, const char c)` is used to save the image with the specified name `f` and `c` indicate the encoding method with a for ASCII and b for binary.

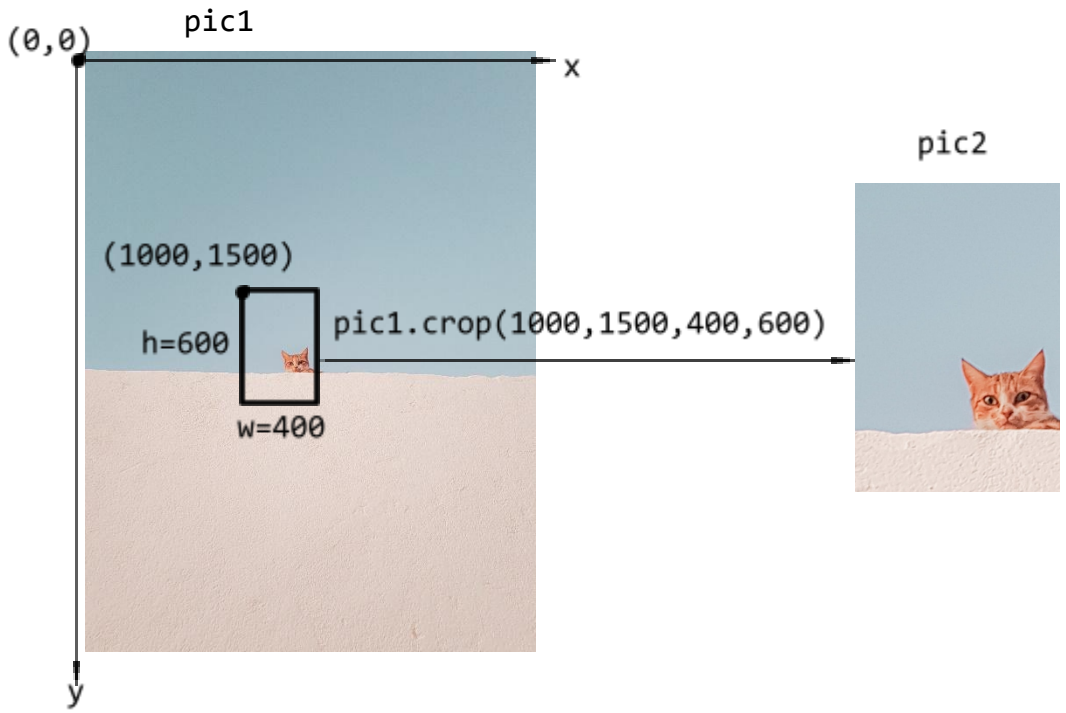
D-02 (20 points)

Give test Q2 in the main function for this problem,

```
    //Q2(20%)
    cout<<"Q2"<<endl;
    ppm pic2 = pic1.crop(1000,1500,400,600);
        cout<<(pic2.saveToFile("t2_1.ppm",  'a')?"write  a  file
t2_1.ppm ":"fail to write a binary file")<<endl;
    pic2 = pic1.crop(600,1350,1000,1000);
        cout<<(pic2.saveToFile("t2_2.ppm",  'b')?"write  a  file
t2_2.ppm ":"fail to write a binary file")<<endl;
```

(4) Please complete overloading `operator=(const ppm &)`.

(5) `crop(int x, int y, int w, int h)` would return an image with width `w` and height `h` started from (x, y).

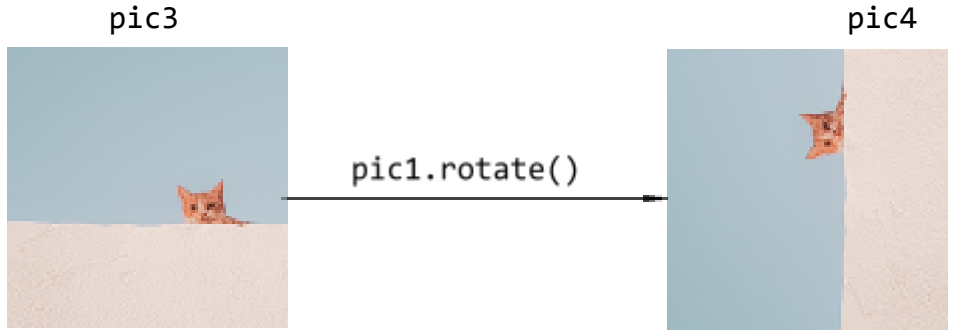


D-03 (20 points)

Given test Q3 in the main function for this problem:

```
    //Q3(20%)
    cout<<"Q3"<<endl;
    ppm pic3 = pic2.resize(10,10);
        cout<<(pic3.saveToFile("t3_1.ppm",  'b')?"write  a  file
t3_1.ppm ":"fail to write a binary file")<<endl;
    ppm pic4 = pic3.rotate();
        cout<<(pic4.saveToFile("t3_2.ppm",  'b')?"write  a  file
t3_2.ppm ":"fail to write a binary file")<<endl;
```

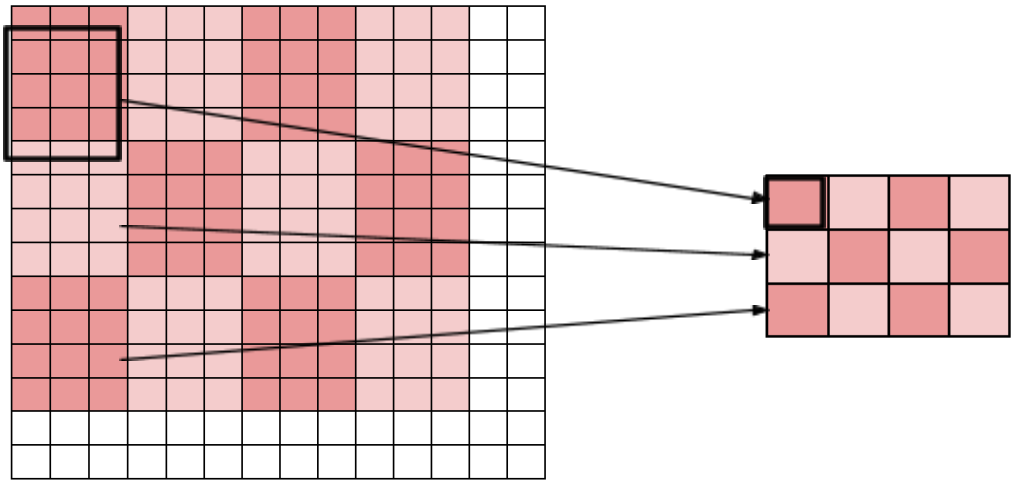
(4) `rotate()` would return an image with the same size but with image rotating counterclockwise.



(5) `resize(int wx, int hx)` would return an image with width and height scales down to `wx` and `hx` times. For example, if the width and height of the original image is 217 and 322, then the width and height of the resized

image is 21 and 16 when `wx` set to `10` and `hx` set to `20`. (Round down to the nearest integer.)

Besides, to resize an image to a smaller size, we must do interpolation to reduce distortion. There are many interpolation method; in this problem, you just need to implement the simplest one: compute the average value of pixels that would compose the same pixels in the new image from the original image.



Take the same example, if the width and height of the original image is 217 and 322 and we want to down scale the width and height to 10 and 20 times seperately, the width and height of the new image is 21 and 16. We can get the new pixel value on the first column and the first row of the resized image from computing the average of first 10 rows and 20 columns on the orignal image.

APPENDIX

There is some methods to check the image you generate.

(1) Use `diff [image1] [image2]` to compare with the image we provide.

(2) Use `tiv [image].ppm`. (`tiv` would compress the image, so do not use it to check the detail of your image.)

(3) Download the image to your computer and use specified image viewer to open it. For Mac OS, please use: `scp -P 3022 [user]@140.113.236.57:[image path] [destination path on your computer]` to transfer the image to your computer. (For more information about `scp`, you can chech here: https://blog.gtwang.org/linux/linux-scp-command-tutorial-examples/.)

Image viewer

Online PPM viewer:
http://www.cs.rhodes.edu/welshc/COMP141_F16/ppmReader.html

IrfanView (Windows only): https://www.irfanview.com/

GIMP: https://www.gimp.org/downloads/

【Problem Set E】(60%)

In this problem, you need to complete a program that can manage products of different companies in a storage room.

In a storage room, it looks like a two-dimensional array. Each element in this room represents a space that can store a product. Each company has a n_1*n_2 space, and companies can place product in their own space only. For example, the below table shows a situation that two companies store their products in the storage room.

x	x			
x				
			x	

In this table, it shows that

1. The size of the storage room is a 5 x 5 array.
2. The size of the first company is a 3 x 2 array whose color is gray. In the grids, other companies can't store their products here.
3. The size of the second company is a 2 x 2 array whose color is black. In the grids, other companies can't store their products there.
4. The n_1*n_2 space which each company allocates is a square or rectangle.
5. The grid can store one product only.
6. The grid which shows character 'x' has stored a product.

Important

- You need to download the files and don't rewrite the file names.
  - /home/share/midterm/pe/StorageSystem.h
  - /home/share/midterm/pe/StorageSystem.cpp
  - /home/share/midterm/pe/main.cpp
  - /home/share/midterm/pe/test.txt
  - /home/share/midterm/pe/cin.txt

- Demo program is in /home/share/pe/demoE-# (# is the index of the program). Please change your working directory to the directory that stores the Q# (# is the number of the question) programs and run the below command.

$ /home/share/midterm/pe/demoE-#

<u>E-01</u> BASICS (5 points)

Please complete the StorageSystem.h and compile it without any error.

In the header file, it is a nested class, and each class that is in it has special features.

1. StorageSystem class
   - Record the size of the storage. (width and height variable)
   - Record all companies which are in the storage room. (companySize and companiesHead variable)
   - Add or Delete the new company. (addCompany function)
   - Display current storage room status. (show function)
2. Company class
   - Record the company's name. (id variable)
   - Record the n_1 and n_2 values. (width and height variable)
   - Record the top left corner position in the n_1*n_2 space. (posX and posY variable)
   - Record all grids that belong to this company. (gridSize and gridsHead variable)
   - Return the grids that are filled in the n_1*n_2 space. (show function)
3. Grid class
   - Record the grid position in the storage room. (posX and posY variable)

Please comply the bellow rules to rewrite the header file.

1. The width variable of StorageSystem class is **20**.
2. StorageSystem object can directly access the private members of Company object.
3. Company object can access the private members of Grid object.
4. Don't add any new functions and rewrite parameters, variables type, or content of origin functions.

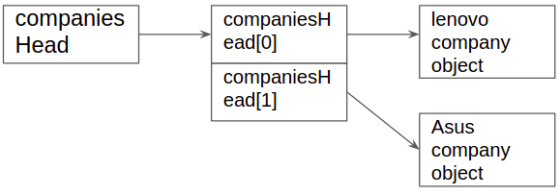
E-02 Dynamic matrix using shallow copy (35 points)

Please complete the StorageSystem.cpp and don't edit the StorageSystem.h header file.

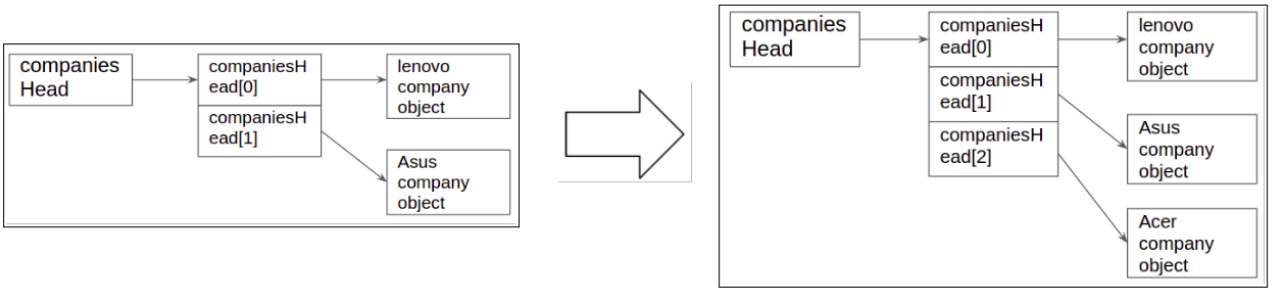
In this problem, you need to first know How the companiesHead or the gridsHead variable work.This part will demonstrate how to add a company or delete a company by using the companiesHead variable.

Add a new company:

If the storage room has record two companies (e.g., lenevo and Asus) information. It shows the below figure.

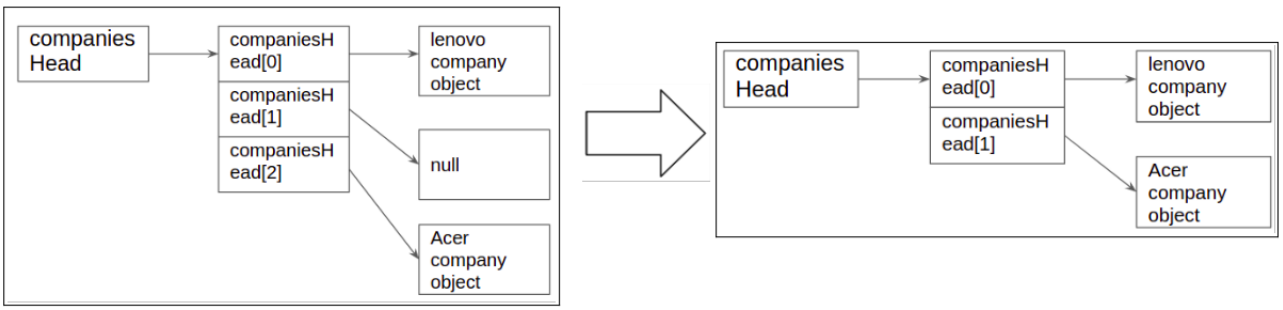


Now, the new company(e.g., Acer) will be added to this storage room. The companiesHead variable will repoint a larger continuous matrix that can store three company object pointers. The new matrix needs to shallow copy the pointers in the old matrix, and then the last element of it stores the new company object address.



Delete a company:

The situation is the same as above. Now, if Asus company will be deleted in this storage room. The companiesHead[1] will delete the Asus company object and then it will be assigned a null value. After it, the companiesHead variable will repoint a smaller continuous matrix that can store two company object pointers. The new



matrix needs to shallow copy the pointers which the value is not null in the old

matrix.

As stated above, how the gridsHead variable work is also the same as the companiesHead variable.

This part will read a text file to create the storage room. The text file's format is that

Line 0: <the height of the storage room>
Line 1: <1's company name> <1's company posX> <1's company posY> <1's company width> <1's company height> <1's product posX> <1's posduct posY> <2's product posX> <2's posduct posY>...
Line 2: <2's company name> <2's company posX> <2's company posY> <2's company width> <2's company height> <1's product posX> <1's posduct posY> <2's product posX> <2's posduct posY>…
Line 3: ...
…
Line N: ...

You don't need to write a parser function because the parser stage is finished in "main.cpp". You only rewrite the content of functions in which the comment is "part2" in "StorageSystem.cpp".

Attention

- You don't need to consider the case that a new company will cover another company. It doesn't occur in demo cases.
- The company or grid that earlier appears in the text file will first store in the matrix.
  (e.g., The first company's index number is zero in the matrix.)
- Use this command to test your code.

  $ ./main test.txt

<u>E-03</u> Dynamic add/delete a company (20 points)

In this part, the program will prompt a user to input a command to show if adding or deleting a company or terminating this program. The commands are that

add command: a <company name> <company posX> <company posY> <company width> <company height> <1's product posX> <1's posduct posY> <2's product posX> <2's posduct posY>…
delete command: d <target company name>
terminate command: CTRL+C

You don't need to write a parser function because the parser stage is finished in "main.cpp". You only rewrite the content of functions in which the comment is "part3" in "StorageSystem.cpp".

Attention

- Use this command to test your code.

```
$ ./main test.txt < cin.txt
```