Name: Ritabroto Ganguly
Roll: 001910501090
OOP Assignment 4

1. Design the class(es) for the following. Each account has account number and balance amount. A list of account is to be maintained where one can add and find account, display information of all accounts. While adding, account number must be unique. Withdraw object has account number (must exist) and amount (will not exceed balance amount of corresponding account). Withdraw object will update the balance of corresponding account in the list. Implement your design. Use friend function wherever required and again, modify your implementation to avoid friend function.

```
#include <iostream>
#include <vector>
using namespace std;
const int accounts = 100;

class account{
  static int ids;
  float balance;
  int id;
  public:
  static vector<account*> accs;
  account(float balance = 0):balance(balance)
{id=ids;accs.push_back(this);ids++;}
  friend ostream& operator<<(ostream& o,const account* x);
  static void show_all_accounts(){
     cout<<"A "<<"B"<<endl;
     for(int i=0;i<accs.size();i++){
        cout<<accs[i]<<endl;
     }
  }
  int get_acc_num(){return id;}
  int get_bal(){return balance;}
  void set_bal(int x){balance = x;}
  static void find_account(int id){
      for(int i=0;i<accs.size();i++)
         if(accs[i]->id==id){
            cout<<"A "<<"B"<<endl;
            cout<<accs[i]<<endl;
```

```
          return;
        }

     cout<<"Account not found"<<endl;
  }
};

int account::ids = 1;
vector<account*> account::accs;
ostream& operator<<(ostream& o,const account* x){
   cout<<x->id<<" "<<x->balance;return o;
}

class Withdraw{
  int amount;
  int acc;
  public:
  void withdraw(int acc,int amt){
     for(int i=0;i<account::accs.size();i++)
        if(account::accs[i]->get_acc_num() == acc){
          if(amt<=account::accs[i]->get_bal())
            account::accs[i]->set_bal((account::accs[i]->get_bal()-amt));
          else
            cout<<"Insufficient balance"<<endl;
          return;
        };

    cout<<"No such account"<<endl;
  }
};

int main()
{
 account x;
 Withdraw w;
 w.withdraw(x.get_acc_num(),100);
 x.set_bal(1000);
 w.withdraw(x.get_acc_num(),100);
 account y;
 account::find_account(2);
 account::show_all_accounts();
 return 0;
}
```

```
Insufficient balance
A B
2 0
A B
1 900
2 0
```

2. Design a COMPLEX class, which will behave like normal integer with respect to
addition,
subtraction,
accepting the value and Displaying the value.

```
#include <iostream>
using namespace std;

template<class T>
class COMPLEX{
  T real;
  T img;
  public:
  COMPLEX(){}
  COMPLEX(T r,T i=0):real(r),img(i){}
  friend ostream& operator<<(ostream& o,const COMPLEX<T>& c){
    o<<c.real<<"+i("<<c.img<<")";
    return o;
  }
  friend istream& operator>>(istream& i,COMPLEX<T>& c){
    cout<<"Real part: ";
    i>>c.real;
    cout<<"Imaginary part: ";
    i>>c.img;
    return i;
  }
  friend COMPLEX<T> operator+(const COMPLEX<T>& a,const
COMPLEX<T>& b){
     COMPLEX<T> c((a.real+b.real),(a.img+b.img));
     return (c);
  }
  friend COMPLEX<T> operator-(const COMPLEX<T>& a,const
COMPLEX<T>& b){
     COMPLEX<T> c((a.real-b.real),(a.img-b.img));
     return (c);
  }
};


int main()
{
 COMPLEX<double> a(1.2,2.3),b;
 cin>>b;
 COMPLEX<double> c = a-COMPLEX<double>(1.3);
```

```
 cout<<c<<endl;
 c = a+b;
 cout<<c<<endl;
 return 0;
}
```

```
Real part: 1.5
Imaginary part: 7
-0.1+i(2.3)
2.7+i(9.3)
```

3. Design an ARRAY class with the following features:
a. Array object may be declared for a specific size and a value for initializing all the elements. If this it is to be assumed as a 0.
b. An array object may be declared and initialized with another object.
c. An array object may be declared and initialized with another array (not the object, standard array as in C language).
Let a and b are two objects:
i. a+b will add corresponding elements.
ii. a=b will do the assignment.
iii. a[I] will return the ith element of the object. iv. a*5 or 5*a will multiply the element with 5.

```
#include <iostream>
#include <cassert>
using namespace std;

template<class T>
class ARRAY{
  T* ptr;
  public:
  const int size;
  ARRAY(int size,int val = 0):size(size){assert(size>0);ptr = new T[size];for(int
i=0;i<size;i++)ptr[i]=val;}
  ARRAY(T* const arr,int size):size(size){assert(size>0);ptr = new T[size];for(int
i=0;i<size;i++)ptr[i]=arr[i];}
  ARRAY(const ARRAY<T>& arr):size(arr.size){
   //cout<<“Copy cst"<<endl;
   assert(size>0);ptr = new T[size];for(int i=0;i<size;i++)ptr[i]=arr.ptr[i];
  }
  ARRAY(ARRAY<T>&& arr):size(arr.size){
   // cout<<"Move cst"<<endl;
```

```
    ptr = move(arr.ptr);
    arr.ptr = nullptr;
  }
  T& operator[](int i) const{
    assert(i<size && i>=0);
    return ptr[i];
  }
  ARRAY<T> operator+(const ARRAY<T>& arr){
     T* p = new T[arr.size];
     for(int i=0;i<arr.size;i++)
        p[i] = (this->ptr[i])+(arr[i]);
     ARRAY<T> x(p,arr.size);
     return x;
  }
  void operator=(const ARRAY<T>& arr){
     assert(this->size>=arr.size);
     for(int i=0;i<arr.size;i++)
        (this->ptr[i]) = (arr[i]);
  }
  template<class T2>
  friend void operator*(T2 x,ARRAY<T>& arr){
    for(int i=0;i<arr.size;i++)
        arr[i] = 5*arr[i];
  }
  template<class T2>
  friend void operator*(ARRAY<T>& arr,T2 x){
     x*arr;
  }
  friend ostream& operator<<(ostream& o,const ARRAY<T>& arr){
      for(int i=0;i<arr.size;i++)
         cout<<arr[i]<<" ";
     return o;
  }
  ~ARRAY(){if(ptr) delete []ptr;}
};

int main()
{
 ARRAY<int> a(10,5);
 cout<<"a= "<<a<<endl;
 ARRAY<int> b(a);
 cout<<"b= "<<b<<endl;
 ARRAY<int> d = a+b;
 cout<<"d= "<<d<<endl;
 ARRAY<int> c = a;
```

```
 cout<<c<<endl;
 return 0;
}
```

```
a= 5 5 5 5 5 5 5 5 5 5
b= 5 5 5 5 5 5 5 5 5 5
d= 10 10 10 10 10 10 10 10 10 10
5 5 5 5 5 5 5 5 5 5
```

4. Design a STRING class, which will have the initialization facility similar to array class. Provide support for
• Assigning one object for another,
• Two string can be concatenated using + operator,
• Two strings can be compared using the relational operators.

```
#include <iostream>
#include <cassert>
#include <cstring>
using namespace std;

class STRING{
  private:
  char* s;
  int size;
  public:
  STRING(char* const c):size(strlen(c)){assert(size>0);s=new
char[size+1];strcpy(s,c);}
  STRING(const char* const c):size(strlen(c)){assert(size>0);s=new
char[size+1];strcpy(s,c);}
  STRING(const STRING& x):size(x.get_size()){assert(size>0);s=new
char[size+1];strcpy(s,x.s);}
  friend ostream& operator<<(ostream& o,const STRING& s) {
     o<<s.s;
     return o;
  }
  int get_size() const {return size;}
  int operator==(const STRING& x){
     if(x.get_size()!=size)
       return 0;

     return strcmp(s,x.s);
  }
  friend STRING operator+(const STRING& a,const STRING& b){
```

```
      //cout<<"+: "<<a<<" "<<b<<endl;
      int sizet = a.get_size()+b.get_size();
      //cout<<"sizet="<<sizet<<endl;
      char *st = new char[sizet+1];
      int i;
      for(i=0;i<(a.get_size());i++)
         st[i] = a.s[i];
      for(int j=0;j<(b.get_size());j++)
         st[i++] = b.s[j];
     st[i] = '\0';
     //cout<<"Final copy st: "<<st<<endl;
     STRING x(st);
     return x;
  }
  void operator=(STRING& x){
      size = x.size;
      delete s;
      s = new char[size+1];
      strcpy(s,x.s);
  }
  ~STRING(){if(s) delete []s;}
};

int main()
{
 char *a = new char[5];
 strcpy(a,"kola");
 STRING x(a);
 cout<<"x="<<endl;
 cout<<x<<endl;
 cout<<x.get_size()<<endl;

 STRING c = x+"ola";
 cout<<"c="<<endl;
 cout<<c<<endl;
 cout<<c.get_size()<<endl;

 cout<<(c=="kolaol")<<endl;

 STRING *y = new STRING(x);
 cout<<"y="<<endl;
 cout<<*y<<endl;
 cout<<y->get_size()<<endl;
 delete y;
 return 0;}
```

```
x=
kola
4
c=
kolaola
7
0
y=
kola
4
```

5. Modify the STRING class so that assigning/initializing a string by another will not copy it physically but will keep a reference count, which will be incremented. Reference value 0 means the space can be released.

```
#include <iostream>
#include <cstring>
#include <cassert>
using namespace std;

class STRING{
  private:
  char* s;
  int *count;
  int size;
  public:
  STRING(char* const c):size(strlen(c)),count(new int(1)){assert(size>0);s=new
char[size+1];strcpy(s,c);}
  STRING(const char* const c):size(strlen(c)),count(new int(1))
{assert(size>0);s=new char[size+1];strcpy(s,c);}
  STRING(const STRING& x):size(x.get_size()),count(x.count)
{assert(size>0);s=x.s;(*count)++;}
  friend ostream& operator<<(ostream& o,const STRING& s) {
     o<<s.s;
     return o;
  }
  int get_size() const {return size;}
  int operator==(const STRING& x){
     if(x.get_size()!=size)
       return 0;

     for(int i=0;i<size;i++)
        if(x.s[i]!=s[i])
```

```
        return 0;
    return 1;
  }
  friend STRING operator+(const STRING& a,const STRING& b){
     //cout<<"+: "<<a<<" "<<b<<endl;
     int sizet = a.get_size()+b.get_size();
     //cout<<"sizet="<<sizet<<endl;
     char *st = new char[sizet+1];
     int i;
     for(i=0;i<(a.get_size());i++)
        st[i] = a.s[i];
     for(int j=0;j<(b.get_size());j++)
        st[i++] = b.s[j];
    st[i] = '\0';
    //cout<<"Final copy st: "<<st<<endl;
    STRING x(st);
    return x;
  }
  void operator=(STRING& x){
    if(x.s==s)return;

    (*count)--;
    if(*count<=0) {delete []s;delete count;}
    s = x.s;
    count = x.count;
    size = x.get_size();
    (*count)++;
  }
  int ptr_count(){return *count;}
  ~STRING(){if(*count<=1) {/*cout<<"deleted: "<<s<<endl;*/delete
count;delete []s;}/*else cout<<"undeleted: "<<s<<endl;*/}
};

int main()
{
 char *a = new char[5];
 strcpy(a,"kola");
 STRING x(a);
 cout<<"x : "<<endl;
 cout<<x<<endl;
 cout<<"size="<<x.get_size()<<endl;
 cout<<"ptr_count="<<x.ptr_count()<<endl;
 cout<<"After STRING z = x : ";
 STRING z = x;
 cout<<"ptr_count="<<x.ptr_count()<<endl;
```

```
 STRING c = x+"ola";
 cout<<"c : "<<endl;
 cout<<c<<endl;
 cout<<"size="<<c.get_size()<<endl;
 cout<<"ptr_count="<<x.ptr_count()<<endl;

 cout<<"c== \"kolaol\" ? "<<(c=="kolaol")<<endl;

 STRING *y = new STRING(x);
 cout<<"y : "<<endl;
 cout<<*y<<endl;
 cout<<"size="<<y->get_size()<<endl;
 cout<<"ptr_count="<<x.ptr_count()<<endl;
 delete y;
 return 0;
}
```

```
x :
kola
size=4
ptr_count=1
After STRING z = x : ptr_count=2
c :
kolaola
size=7
ptr_count=2
c== "kolaol" ? 0
y :
kola
size=4
ptr_count=3
```

6. For each student roll, name and phone number are to be maintained. For each subject store subject code, name. A student may choose number of subjects from a list of subjects. A subject may be chosen by number of students. Given a roll number system must able to display the subjects chosen by the student and also be able to display the students corresponding to a subject code. Design the classes and implement the system.

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

class student;
class Key;

class subject{
  static int codes;
  int code;
  string name;
  vector<student*> students;

  public:
  subject(string name):code(codes),name(name){codes++;}

  int get_code(){return code;}

  string get_name(){return name;}

/*Extra key parameter used following the Passkey idiom to restrict access of
this function*/
  void add_student(const Key&,student *stu){students.push_back(stu);}

  friend ostream& operator<<(ostream& o,const subject& s){
     o<<s.code<<"\t"<<s.name<<endl;
     return o;
  }

  vector<student*> get_students(){return students;}
  void show_students();
};
int subject::codes = 1;
vector<subject> subjects;

class student{
  static int rolls;
  int roll;
  string name;
  int number;
  vector<subject*> subjects;

  public:
```

```
  student(string& name,int number):roll(rolls),name(name),number(number)
{rolls++;}

  int get_roll(){return roll;}

  int get_number(){return number;}

  string get_name(){return name;}

  void set_number(int newNumber){number=newNumber;}

  friend ostream& operator<<(ostream& o,const student& s){
     o<<s.roll<<"\t"<<s.name<<endl;
     return o;
  }

  void add_subject(int);

  vector<subject*> get_subjects(){return this->subjects;}
  void show_subjects(){for(int i=0;i<this->subjects.size();i+
+)cout<<*subjects[i];}
};
int student::rolls = 1;
vector<student> students;

void subject::show_students(){for(int i=0;i<students.size();i+
+)cout<<*students[i];}

/* Key class for Passkey Idiom */
class Key{
  Key(){} // default ctor private to restrict access to Key
  Key(const Key&){} // copy ctor private to restrict access to Key

  // grant access to add_subject() method of student to create object of Key
  friend void student::add_subject(int);
};

void student::add_subject(int subs){
    cout<<"Enter subject name for Name:"<<this->name<<", Roll:"<<this-
>roll<<" : ";
    string x;
    getline(cin,x);
    subject *sub = nullptr;
    for(int i=0;i<subs;i++)
      if(::subjects[i].get_name()==x)
```

```
        sub = &(::subjects[i]);

    if(sub!=nullptr){
      subjects.push_back(sub);
      sub->add_student(Key(),this);
    }else{
       cout<<x<<" subject is not available"<<endl;
    }
}

//string x[] = {"English","Bengali","Hindi"};
//string y[] = {"rg","sg"};

int main()
{
 int subs = 3,stus = 2;
 cout<<"Enter number of subjects: ";
 cin>>subs;
 cout<<"Enter number of students: ";
 cin>>stus;
 cin.ignore();

 for(int i=0;i<subs;i++){
    // subjects.push_back(subject(x[i]));
   string s;
   cout<<"Enter subject name: ";
   getline(cin,s);
   subjects.push_back(subject(s));
 }
 for(int i=0;i<stus;i++){
     //students.push_back(student(y[i],100*(i+1)));
     string s;
     cout<<"Enter student name: ";
     getline(cin,s);
     int x;
     cout<<"Enter student number: ";
     cin>>x;
     cin.ignore();
     students.push_back(student(s,x));
     for(int j=0;j<=i;j++)
       students[i].add_subject(subs);
 }

 for(int i=0;i<subs;i++){
    cout<<subjects[i];
```

```
    subjects[i].show_students();
  }
  cout<<"---------------------------------"<<endl;
  for(int i=0;i<stus;i++){
    cout<<students[i];
    students[i].show_subjects();
  }
  return 0;
}
```

```
Enter number of subjects: 3
Enter number of students: 2
Enter subject name: eng
Enter subject name: beng
Enter subject name: hin
Enter student name: r g
Enter student number: 13930139
Enter subject name for Name:r g, Roll:1 : eng
Enter student name: s g
Enter student number: 1341414
Enter subject name for Name:s g, Roll:2 : beng
Enter subject name for Name:s g, Roll:2 : hin
1       eng
1       r g
2       beng
2       s g
3       hin
2       s g
---------------------------------
1       r g
1       eng
2       s g
2       beng
3       hin
```