I hereby pledge that I will strictly adhere to academic integrity codes and the work done on this examination is solely my own and I will not receive/give any help from/to anybody or source during this examination.

Süleyman Gölbol
1801042656
CSE 241-501 Midterm Examination
~~SG~~.

1) a) What is a conversion constructor? What is keyword explicit? How are they related?

b) What is compilation unit? What are unnamed namespace? How are they related?

a) For example, we have a class that has 2 private members. int a and char b. Conversion constructors are the constructors that completes the missing part. For example we have class 'C'.

```
C c1 = 'b' ; C c2 = 5;
```

The conversion constructor allows us not to implement both a and b. So we can create objects like above.
When we put explicit keyword before the constructor, that means "Don't allow to use conversion constructor." So the code above fails. When we create objects, we have to use our normal constructors.

b) Compilation unit is the group of unit that we compile or use at the same time. For example class.h and class.cpp is together a compilation unit.
when we use unnamed namespace; we cannot access the members in unnamed namespace from outside or other compilation units. So we don't have to worry about if someone uses something unnecessary in my class.

Süleyman Gölbol
1801042656
CSE 241-501 Midterm Examination
~~56~~.

Q2) Implement a global C++ function that takes an array of Cell objects that represent the cell of a hex game. Your function will find the cell with the smallest row number and return it. Assume that Cell class is defined with all necessary setters and getters. write only one function.

```
Hex:: Cell function (Hex::Cell hexCells[][26]){
    bool flag=false;
    Cell c;     // 26 is because of there are 26 letters.
    for (int i=0; i<26; i++){
        for(int j=0; j<26; j++){
            if(hexCells[i][j].getStatus()=='x'
               || hexCells[i][j].getStatus()=='o'){
                flag = true;
                c.setX(j);
                c.setY(i);
                c.setStatus(hexCells[i][j].getStatus());
                break;
            }
        if(flag==true) break;
        } //End of seond for loop.
    }//End of first for loop
    return c;
} // End of scope
```

Süleyman Gölbol
1801042656
CSE 241-501 Midterm Examination

Q3) Design and implement a C++ class to represent a person. Your class will have the following functions as well as any other functions necessary. (setters, getters, etc)

A constructor that takes all parameters (name, last name, age, gender)
A function that returns the number of existing person objects,
Overloaded stream insertion operator (cout << person) will print details
Overloaded operator == and operator != for comparing persons.
Overloaded pre and post increment operators that increment age by 1.
Overloaded operator < that compares Person ages. Perform error checks.
Don't write header file.

```
using namespace std;
class Person {
    private:
        string name;
        string lastname;
        int age;
        char gender; // 'M' for male, 'F' for female
        static int existingPersons;
    public:
        Person(string name2, string lastname2, int age2, char gender2);
        static int getExistingPerson();
        static void setExistingPerson();
        friend ostream& operator<<(ostream& os, const Person& p);
        bool operator==(const Person& p1, const Person& p2);
        bool operator!=(const Person& p1, const Person& p2);
        Person& operator++(); // Pre
        Person& operator++(int ignore); // Post
        bool operator<(const Person& p1, const Person& p2);
        inline string getName() { return name; }
        inline string getLastName() { return lastname; }
        inline int getAge() { return age; }
        inline char getGender() { return gender; }
        inline void setName(string name2) { name = name2; }
        inline void setLastName(string lastname2) { lastname = lastname2; }
        inline void setAge(int a) { age = a; }
        inline void setGender(char g) { gender = g; }
};
```

```
Person::Person(string name2,string lastname2, int age2, char gender2){
    if(gender!='F' || gender!='M' || age<0){cout<<"Error"; return
    name = name2;
    lastname = lastname2;
    age = age2;
    gender = gender2;

}
int getExistingPerson(){        // Static
    return existingPersons;
}
void setExistingPerson(){       // Static
    existingPerson++;
}
friend ostream& operator<<(ostream& os, const Person& p){
    os<<name<<endl;
    os<<lastname<<endl;
    os<<age<<endl;
    os<<gender<<endl;
    return os;
}
bool operator==(const Person& p1, const Person& p2){
    if(p1.name==p2.name && p1.lastname==p2.lastname
    && p1.age==p2.age && p1.gender==p2.gender)

    return true;
    return false
}
```

Süleyman Gölbol
1801042656
CSE241-501 Midterm Examination
SG.

Q4) Define and implement a C++ class to represent a vector of doubles, such as vector<double>. Your new class will work with doubles only and it will have the following features.

Uses dynamic memory for storing doubles.
overloaded operator[] and operator()
functions push_back, pop_back, capacity and size
Creates a new namespace
has a seperated interface and implementation
overloaded operator += appends another vector to the end of this vector.
uses keyword decltype and auto
any other functions needed

```
namespace GTU{
class dVector{
        private:
            double *d;
            int mySize = 0;

        public:
            dVector(double *d, int size2);
            double operator[](int index);
            friend ostream & operator<<(ostream& os, dVector object);
            DVector& operator+=(const DVector& o1, const DVector& o2);
             void push_back(double d);
             void pop_back();
            inline int size(){ return mySize;}
             inline int capacity(){ return size*8;}

            ~dVector();
            dVector(const dVector & object); // copy constructor

};
```

Süleyman Gölbol
180104256
SG.

```
dVector::dVector(double *d2, int size2){
    d = new double[size2];
    for(int i=0; i<size2; i++){
        d[i] = d2[i];
    }
    mysize = size2;
}

void dVector::push_back(double d2){
    double temp = new double[mysize+1];
    for(int i=0; i<mysize; i++)
        temp[i] = d[i];
    temp[i+1] = d2;
    delete[] d;
    d = nullptr;
    d = new double[mysize+1];
    mysize += 1;
    for(int i=0; i<mysize; i++)
        d[i] = temp[i];
    delete[] temp;
}

void dVector::pop_back(){
    double temp = new double[mysize-1];
    for(int i=0; i<mysize-1; i++)
        temp[i] = d[i];
    delete[] d;
    d = nullptr;
    d = new double[mysize-1];
    mysize -= 1;
    for(int i=0; i<mysize; i++)
        d[i] = temp[i];
    delete[] temp;
}
```

Süleyman Gölbol
1801042656

```
double operator [] (int index) {

        if(index > mysize){
            cout << "Error\n";
            return d[0];
        }
        return d[index];
}

friend ostream& operator<< (ostream &os, dVector object) {
        for(int i=0; i< object.mysize; i++){
            os << object.d[i];
            os << endl;
        }
        return os;
}

dVector& operator +=( const DVector& o1, const DVector& o2) {
    double *temp=new double[o1.mysize +o2.mysize];
    for(int i=0;i<o1.mysize; i++)
        temp[i] = o1.d[i];
    for(int i=0; i<o2.mysize ; i++)
        temp[o1.mysize +i] = o2.d[i];
    DVector myd( temp, o1.mysize +o2.mysize);
    return myd;
}

dVector::~dVector(){
        delete [] d;
}

} //End of namespace GTU
```