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

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 any body or source during this examination.

a) What is slicing problem in C++? Does Java has slicing problem?

b) What are the differences between move constructors and copy constructors?

a) In C++, if we have a base class object and derived class object; if we assign baseObject = derivedObject; because of base class doesn't have functions that derived class have, when we assign it, object is slicing.

No, Java doesn't have slicing problem. Because in Java, opposite to C++, when we assign it, it is with references. So, this problem doesn't occur in Java.

b) Copy constructors are for lvalues. When it returns, we assign the returned value to an lvalue. But with move constructor, it doesn't use &, it uses && and that is for rvalues. Inside of move castructor, the argument object in the parameter if it changes in the implementation, it is not important, because it doesn't return a reference. But in copy constructor it is important because it uses & reference. So in parameter we write const to show that it shouldn't and cannot change.

Süleyman Gölbol
1801042656
CSE 241-501 Final Examination

a) Implement a global templated C++ function that takes an array as a parameter. Your function returns the median of the array without copying or modifying the array.

b) Write a static Java method that takes integer as a parameter and returns a String that contains the binary representation of this integer. If the integer is 21, the returned string is "10101"

a)

```
template < typename T >
T median (const T& array[], int size){

        int howMany [size]={0};///it will keep this index's value how many
                                  // times used.

        for(int i=0, i<size; i++)
            for(int j=0, j<size; j++)
                    if (array [i]== array[j])
                        howMany[i] ++;


        int which= 0;
        for (int i=1 ; i< size; i++)
                if ( howMany[i]> howMany [which])
                    which= i;

        return howMany [which];
}
```

b)

```
static String binary (int number){
        String s="";
        while (number / 2){
                s = number%2 +s; // In string places are important for +
                number = (int) number /2 ;
        }
        return s;
}
```

Süleyman Gölbol
1801042656 SG.
CSE 241-301 Final Examination

3) a) Design an implement a Comparable Java class to represent a rational number such as 3/2. Your class will have the following methods as well as any other methods necessary (setters, getters)

- A constructor takes all paramaters (numerator, denumerator)

A function returns number of existing Rational objects,

Override toString method

Methods for adding and multiplying Rational objects

Methods will throw exceptions if there are problems.

b) write another class to test your Rational class including the exceptions.

```
Public class Rational implements Comparable throws Exception {
    private double d;
    Private static int existingObjects;
    private int numer, denum;
    Public Rational (int numerator, int denumerator)){
        numer= numerator;
        denum=denumerator;
        d=(double) numerator/denumerator;
        existingObject ++;
    }
    public static int getExistingObjects (){
        return existingObjects;
    }
    @override
    public String toString(){
        String s= numer + "/" + denum + "=" + d;
        return s;
    }
    public Rational sum(Rational r1, Rational r2){
        Rational r;
        try{
            r.d= r1.d + r2.d;
            if (r1.denum ==r2.denum) {r.numer = r1.numer+ r2.numer;
            else{                    r.denum = r1.denum; }
                r.numer= r1.denum * r2.numer + r1.numer * r2.denum;
                r.denum =r1.denum * r2.denum;
            }
        catch(Exception& e){
            System.out.println ( e.getMessage() + " caught.");
        }
        return r;
```

```
public double getD(){
        return d;
}
 public void setD(double d){
        this.d = d;
 }

 public Rational multiply(Rational r1, Rational r2){
        Rational r;
        try{
            r.d= r1.d * r2.d;
            r.numer = r1.numer * r2.numer;
            r.denum= r1.denum * r2.denum;
        }
        catch(Exception & e){
            System.out.println(e.getMessage() + " caught");
        }

        return r;
 }
```

Test.java

```
public class Test{
    public static void main(String args[]){
        existingObjects = 0; // Initialization
        Rational r1= new Rational();
        r1.setD(56.7);
        Rational r2 = new Rational();
        r2.setD(41.3);
        r1= r1.sum(r1,r2);
    }
}
```

Süleyman Gölbol
1801042656
CSE241-501 Final Exam [illegible]

4) Set is a templated abstract base C++ class to present a set. It defines (but not implements) regular set functions; add, contains Element intersection (kesişim) and size, SetA, SetC are derived concrete classes that implement all these functions. Set A is an adapter class and uses [illegible] STL classes to keep the elements. SetC uses regular C arrays to keep the set elements.

a) write the class definition for Set class.

b) Design and implement SetA class

c) Design and implement SetC class

d) write a Main function to test your classes.

a)

```
template <typename T>
class Set{
    private:
        int size1;

    public:
           template<typename T>
        virtual void add( const T& var) = 0 ;
           template<typename T>
        virtual bool containsElement( const T& var) = 0 ;
        virtual Set intersection(const Set& s1, const Set& s2) = 0 ;
        virtual int size() const = 0 ;
        Set(); //constructor

};
```

b)

```
template <typename T>
class SetA : public Set{
    private:
        std::Set<T>mySet ; // from STL

    public:
        template <typename T>
        void add(const T& var) override;
        template <typename T>
        bool containsElement(const T& var) override;
        SetA<T>intersection(const Set& s1, const Set& s2) override;
        int size() override;

        SetA() ;

        void printSet() const;
        T get(index i);
};
```

Süleyman Gölbol
1801042656
CSE 241-501

Final
Examination

```
template <typename T>
void SetA<T>::add(const T& var){
        mySet.push(var);

}template <typename T>
bool setA<T>::containsElement(const T& var){
      bool b;
      b=find(mySet.begin(), mySet.end(), var); //from STL
      return b;
 {
  template <typename T>
Set SetA<T>::Intersection (const T& s1, const T& s2){
        std::Set newSet;
        for (int i=0 ;i< s1.size() ; i++)
                for (int j=0 ; j< s2.size() ; j++)
                        if ( s1.containsElement (s1.get(i)) && s2.containsElement(s2.get(j)))
                                newSet.add( s1.get(i));

        return newSet;
{

int SetA<T>::size(){
                return size1;
   {

SetA<T>:: SetA() : Set()
        { /* Empty */}
```

Süleyman Gölbol
1801042656
CSE 241-501 Final Examination

```
c) template <typename T>
   class setC : public Set {
          private :
             T* myset;
          public :
             void add ( const T & var ) override;
             bool containsElement (const T& var) override;
             SetC intersection (const Set& s1, const Set& s2) override;
             int size () override;
             setC ();
             ~ setC();
             SetC <T> setC ( const setC & s);
             Set C <T> operator= ( const SetC & s);
             void printSet() const;
             T get(index i);

   }

template <typename T>
   void setC<T>:: add (const T& var){
          T* temp = new T[ size1+1] ;
          for(int i=0; i< size1; i++)
               temp[i] = myset [i];

          temp [size] = var;
          delete[] myset;    size1++; myset new myset[size1];
          for (int i=0 ; i< size1; i++)
               mySet[i] = temp[i];


   }
```