CSE 241-501 Final Examination Muhammed Bedir

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.

UCUCAY
19010426937 MUcay

Set is a templated abstract base C++ class to present a set

It defines (not implement) regular set functions - add
- containsElement
- intersection (Kesisim)
- size

SetA and SetC are derived concreate classes that implements all these functions. SetA is an adaptor class and uses suitable STL classes to keep the elements. Set C uses regular C arrays to keep the set elements.

- Write the for Set class
- Design and implement SetA class
  " " " SetC clas
- Write a main to test your classes

```
#ifndef Set
#define Set
  template<class T>
  class Set{
      public:
        virtual bool add(T element) =0;
        virtual bool containsElement(T element) const =0;
        virtual void intersection(Set *s) =0;
        virtual int setSize() const =0
        virtual void setSize() =0;
        virtual ~Set(){}

      private:
        int size
};
#endif
```

```
#ifndef SetA
#define SetA
#include <vector>

template<class T>
class SetA : public Set<T> {  private: vector<T> buffer;
public:
    SetA();
    ~SetA();
    bool add(T element) {
        if (!contains(element)) {
            buffer.push_back(element);
            return true;
        }
        return false;
    }
    bool contains(T element) {
        for (int i=0; i < buffer.size(); ++i){
            if( buffer[i] == T){
                return true;
            }
        }
        return false;
    }
    int getSize() { return buffer.size(); }
    void setSize(int size) {
        vector<T> tmp = buffer;
        vector<T> tmp2(size);
        buffer = tmp2;
        for(int i=0; i< buffer.size(); ++i
            buffer[i] = tmp[i];
    }
    void intersection(Set* s) {
      //I don't know implementation
    }
};
#endif//SetA
```

```
#ifndef SetC
#define SetC

template<class T>
class SetC : public Set<T> {
    private:
        T* buffer;
        int used;
    public:
        SetC() {}
        SetC(int size) : size(size), used(0) {
            buffer = (T*) malloc(sizeof(T)* size);
        }
        SetC(SetC& setc) : size(setc.size), used(setc.used) {
            buffer = (T*)malloc(sizeof(T)*size);
            for(int i=0; i<used; ++i) {
                buffer[i] = setc.buffer[i];
            }
        }
        ~SetC() { delete [] buffer; }
        SetC operator = (SetC& setc) {
            if(&setc != this) {
                T* tmp = (T*) malloc(sizeof(T)*setc.size);
                size = setc.size;
                used = setc.used;
                for(int i=0; i<used; ++i)
                    tmp[i] = setc.buffer[i];
                delete [] buffer;
                buffer = tmp;
            }
            return *this;
        }
```

```
// SetC devam
int getSize() { return size }
int setSize(int size) {
    this.size = size;
}
void intersection(Set *s) {
    // I don't know Implementation
}
bool containsElement(T element) const {
    for (int i=0; i<used; ++i) {
        if (buffer[i] == element)
            return true;
        return false;
    }
}
bool add(T element) {
    if(used < size)
        buffer[used] = element;
    else { cout << " This array full "<<endl;
    }    return false;
    used++;
    return true;
}
} // End of Class SetC #endif // SetC

int main() {
    SetA setA<int>;
    setA.add(5); setA.add(3);
    if (setA.containsElement(4)) cout<< "true"<<endl
    SetC setC<int>(9);          SetC setC2<int>; setC2.setSize(7);
    setC.add(7); setC.add(5);
    if (setC.contains(3)) cout<< "true"<<endl
    setA.intersection(&setC);
    cout << setA.getSize();     return 0;
}
```