Lab 06 (Dated 23-10-2020)

1. Set_Operator_Overloading

Create class Set with character type elements. Keep array dynamic to grow size, whenever required. Write parameterized constructor with size as parameter with default value 5. Create copy constructor and assignment operator without code duplication. Write destructor as well. Overload operators:

1. "+" to take union of two sets into new set
2. "*" to print Cartesian product of two sets
3. "-" operator to produce a new set with all elements of first set not existing in second set
4. "+" operator to add new element in set
5. "-" to remove element from set
6. == operator returns true if all the elements of sets are common
7. != operator returns true if one or more elements are not common
8. << operator to print elements of set, if set is empty print "EMPTY SET"
9. Read and add elements in the set
10. Assign one set to another set

In main create Set type array of size 5. Perform operations according to the description given in input format. This description is not part of actual input.

Input Format

15 (Number of operations)
8 2 (Print set 2)
9 2 3 A B C (Read 3 elements in set 2)
8 2 (print set 2)
9 3 3 C B A (Read 3 elements in set 3)
8 3 (Print set 3)
6 2 3 (Call == operator for set 2 and 3 and print TRUE or FALSE)
7 3 2 (Call != operator for set 3 and 2 and print TRUE or FALSE)
4 3 D (Add element D in set 3)
4 3 E (Add element E in set 3)
1 2 3 1 (Find union of set 2 & 3 and store result in set 1)
8 1 (Print set 1)
7 1 3 (Call != operator for set 1 and 3 and print TRUE or FALSE)
7 1 2 (Call != operator for set 1 and 2 and print TRUE or FALSE)
10 4 2 (Assign set 2 to set 4)
2 2 4 (Generate and print Cartesian product of set 2 and 4)

Output Format

EMPTY SET
A B C
C B A
TRUE
FALSE
A B C D E
FALSE
TRUE
{(A, A),(A,B),(A,C),(B,A),(B,B),(B,C),(C,A),(C,B),(C,C)}

Solution:

```
#define NOTFOUND -1

class Set{
    int cSize, size;
    char *e;
    void copy(const Set &s){
        size = s.size;
        e = new char[size];
        copyElements(s);
    }
    void copyElements(const Set &s){
        for (int i=0;i<s.cSize;i++)
            e[i] = s.e[i];
        cSize = s.cSize;
    }
    int isExist(const char ELEMENT) const{//return index of element, otherwise
-1
        for (int i=0;i<cSize;i++)
            if (e[i] == ELEMENT)
                return i;
        return NOTFOUND;
    }
public:
    Set(int size=5){
        if (size<0)    size=5;
        this->size = size;
        cSize = 0;
        e = new char[size];
    }
    Set(const Set &s){
        copy(s);
    }
    Set& operator = (const Set &s){
        delete []e;
        copy(s);
        return *this;
    }
    Set operator + (const Set &s){
        Set newSet(cSize+s.cSize);
        newSet.copyElements(*this);
        for (int i=0;i<s.cSize;i++)
            if (newSet.isExist(s.e[i])==-1)
                newSet.e[newSet.cSize++] = s.e[i];
        return newSet;
    }
    Set operator - (const Set &s){
        Set newSet(cSize);
        for (int i=0;i<cSize;i++)
            if (s.isExist(e[i])==NOTFOUND)
                newSet.e[newSet.cSize++] = e[i];
        return newSet;
```

```
    }
    bool operator + (const char ELEMENT){
        if (cSize==size)                return false;
        if (isExist(ELEMENT)!=NOTFOUND)   return false;
        e[cSize++]=ELEMENT;
        return true;
    }
    bool operator - (const char ELEMENT){
        if (cSize==0)               return false;
        int pos=isExist(ELEMENT);
        if (pos==-1)   return false;
        for ( ;pos<cSize;pos++)
            e[pos] = e[pos+1];
        cSize--;
        return true;
    }
    bool operator == (const Set &s) const{
        int i;
        for (i=0;i<s.cSize;i++)
            if (isExist(s.e[i])==-1)
                return false;
        for (i=0;i<cSize;i++)
            if (s.isExist(e[i])==-1)
                return false;
        return true;
    }
    bool operator != (const Set &s) const{
        return !(*this==s);
    }
    void operator * (const Set &s ){
        int i,j;
        cout <<'{';
        for (i=0;i<cSize;i++)
            for (j=0;j<s.cSize;j++){
                if (i!=0 || j!=0)    cout << ',';
                cout << '(' << e[i] << ',' << s.e[j] << ')';
            }
        cout << "}\n";
    }
    Set& read(const int COUNT){
        char element;
        for (int i=0;i<COUNT;i++){
            cin >> element;
            if (isExist(element)==-1)
                e[cSize++]=element;
        }
        return *this;
    }
    friend ostream& operator << (ostream &, const Set&);
};
ostream& operator << (ostream &out, const Set &s ){
    if (s.cSize==0){
```

```
        out << "EMPTY SET\n";
        return out;
    }
    for (int i=0;i<s.cSize;i++)
        out << s.e[i] << ' ';
    out << '\n';
    return out;
}
int main() {
    Set s[5];
    int s1, s2, s3, operationNo, operationCount, count;
    char element;
    cin >> operationCount;
    for (int i=0;i<operationCount;i++){
        cin >> operationNo;
        if (operationNo==1){
            cin >> s1 >> s2 >> s3;
            s[s3-1] = s[s1-1] + s[s2-1];
        }
        else if (operationNo==2){
            cin >> s1 >> s2;
            s[s1-1] * s[s2-1];
        }
        else if (operationNo==3){
            cin >> s1 >> s2 >> s3;
            s[s3-1] = s[s1-1] - s[s2-1];
        }
        else if (operationNo==4){
            cin >> s1 ;
            cin >> element;
            s[s1-1] + element;
        }
        else if (operationNo==5){
            cin >> s1 >> element;
            s[s1-1] - element;
        }
        else if (operationNo==6){
            cin >> s1 >> s2;
            if (s[s1-1] == s[s2-1])   cout << "TRUE\n";
            else                      cout << "FALSE\n";
        }
        else if (operationNo==7){
            cin >> s1 >> s2;
            if (s[s1-1] != s[s2-1])   cout << "TRUE\n";
            else                      cout << "FALSE\n";
        }
        else if (operationNo==8){
            cin >> s1 ;
            cout << s[s1-1] ;
        }
        else if (operationNo==9){
            cin >> s1 >> count;
```

```
            s[s1-1].read(count);
        }
        else if (operationNo==10){
            cin >> s1 >> s2;
            s[s1-1] = s[s2-1];
        }
    }
    return 0;
}
```