Quinn Roemer

CISP - 440

Assignment 4

10/11/2018

# Part 0 - Set Implementation

## Description:

The goal for this assignment was to write code that would be able to perform a given range of set operations. I was to implement several functions that had yet to be completed. After getting the code to work I was to perform the provided operations on a number of sets.

## Operations to perform:

(A ∪ B) ∩ C
A ∪ (B ∩ C)
~(A ∩ B)
~A ∪ ~B
A - B
PowerSet(A)
Bool A ⊆ B
Bool A ⊂ B

**Plus 2 of my own design:**

|A ∪ ~(B - C)|
Bool (A ∩ B) ⊆ A

## Note:

The code listed in this document is only for the small universe. Since the code is very similar between different universes sizes, this code will be the only version listed.

## Source Code:

```
//Written by Quinn Roemer, based on code by Professor Ross.
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>

using namespace std;

#pragma warning( disable : 4996)
#pragma warning( disable : 4244)

//Small Universe
char Universe[8][10] = { "Bat", "Cat", "Chimp", "Dog", "Fish", "Liger", "Snake", "Turtle" };
```

```
//Large Universe
char BigUniverse[32][20] = {
    "Bat", "Cat", "Chimp", "Dog", "Fish", "Liger", "Snake", "Turtle",
    "Bear", "Dragon", "Horse", "Wolf", "Rat", "Gerbil", "Rabbit", "Monkey",
    "Donkey", "Llama", "Zebra", "Hippopotamus", "Rhinoceros", "Gecko", "Frog", "Sloth",
    "Deer", "Kangaroo", "Gorilla", "Alligator", "Panda", "Squirrel", "Duck", "Platypus" };

typedef unsigned char set;
typedef unsigned long int set32;

//Prints out a set in set-sequence notation
void printSet(set A)
{
    printf("{ ");

    bool commaflag = false;
    int i = 0;
    unsigned char mask = 0x80;
    for (; mask; mask >>= 1, i++) {
        if (mask & A)
        {
                if (commaflag) printf(", ");
                printf(Universe[i]);
                commaflag = true;
        }
    }
    printf(" }");
}

//Prints each bit of a byte
void print8bits(unsigned char x)
{
    for (unsigned char mask = 0x80; mask; mask >>= 1) {
        if (mask & x)
                printf("1");
        else
                printf("0");
    }
}

//Inserts an element of the universe into a set
void insert(set & A, char str[])
{
    // get a unique hash for each string
    int hash = (str[0] + str[2]) % 20;

    int g[20] = { 6, -1, 0, 1, -1, 4, -1, -1, -1, -1, -1, 3, 2, -1, -1, -1, -1, -1, 7, 5 };

    int index = g[hash];
```

```
    // make a mask
    set mask = 0x80 >> index;

    // insert this element
    A = A | mask;
}

//Calculates base raised to the power exp
int my_pow(int base, int  exp)
{
    int x = 1;
    for (int i = 0; i < exp; i++)
        x *= base;

    return x;
}

//==========================
//=== My code starts here ===
//==========================

//This function computes the union of two passed sets.
set Union(set A, set B)
{
    return A | B;
}

//This function computes the intersection of two passed sets.
set Intersection(set A, set B)
{
    return A & B;
}

//This function returns the complement of a passed set.
set Complement(set A)
{
    return ~A;
}

//This function returns the difference two sets.
set Difference(set A, set B)
{
    return Intersection(A, Complement(B));
}

//This function computes the cardinality of a set.
int Cardinality(set A)
{
    //Mask starts at 1000 0000.
    //Note, this will have to be changed for the bigUniverse.
```

```
    int mask = 0x80;
    int card = 0;

    //Iterating through set to find all 1's. Note, iterates to the bit size of char passed.
    for (int count = 0; count < sizeof(A) * CHAR_BIT; count++)
    {
        //If a 1 is found, card will be incremented.
        if (A & mask)
        {
                card++;
        }
        mask >>= 1;
    }
    return card;
}

//This function computes all possible power sets of a given set.
void printPowerSet(set A)
{
    int cardP = my_pow(2, Cardinality(A));
    int exponent = Cardinality(A) - 1;

    set temp;
    set temp2;
    set result = 0;

    int mask = 0x80;

    cout << "Powerset(A): " << endl;

    //This loop generates all possible powerSets.
    for (int count = 0; count < cardP; count++)
    {
        temp = count;

        //This loop iterates through the entire char placing the bits in the correct location.
        for (int index = 0; index < sizeof(A) * CHAR_BIT; index++)
        {
                //If the mask & A = 1 this will execute.
                if (mask & A)
                {
                        //Setting char to correct bit power.
                        temp2 = my_pow(2, exponent);

                        //If the correct location in temp is 1 this will execute.
                        if (temp & temp2)
                        {
                                //Set the result to the OR of mask.
                                result = result | mask;
                        }
```

```
                    exponent--;
                }
                mask >>= 1;
            }
            //Printing the result.
            printSet(result);
            cout << endl;

            //Resetting the necessary variables for the next iteration.
            result = 0;
            mask = 0x80;
            exponent = Cardinality(A) - 1;
        }
    }

    //This function returns true if a subset is passed. Else false.
    bool IsSubset(set ASubset, set ASet)
    {
        if ((ASet | ASubset) == ASet)
        {
            return true;
        }
        return false;
    }

    //This function returns true if a proper subset is passed. Else false.
    bool IsProperSubset(set ASubset, set ASet)
    {
        if (ASet == ASubset)
        {
            return false;
        }
        else if ((ASet | ASubset) == ASet)
        {
            return true;
        }
        return false;
    }

    //Main function to execute.
    void main(void)
    {
        set A = 0;

        insert(A, "Cat");
        insert(A, "Dog");
        insert(A, "Fish");
        printf("Set A: ");
        printSet(A);
```

```
set B = 0;
insert(B, "Cat");
insert(B, "Dog");
insert(B, "Liger");
printf("\nSet B: ");
printSet(B);

set C = 0;
insert(C, "Dog");
insert(C, "Liger");
insert(C, "Snake");
insert(C, "Turtle");
printf("\nSet C: ");
printSet(C);
cout << endl;

set D = 0;

cout << "\n(A union B) Intersection C: ";
D = Intersection(Union(A, B), C);
printSet(D);
cout << endl;

cout << "\nA union (B intersection C): ";
D = Union(A, Intersection(B, C));
printSet(D);
cout << endl;

cout << "\nComplement (A intersection B): ";
D = Complement(Intersection(A, B));
printSet(D);
cout << endl;

cout << "\nComplement A union complement B: ";
D = Union(Complement(A), Complement(B));
printSet(D);
cout << endl;

cout << "\nA difference B: ";
D = Difference(A, B);
printSet(D);
cout << endl << endl;

printPowerSet(A);

if (IsSubset(A, B))
    cout << "\nA is a subset of B" << endl;
else
    cout << "\nA is not a subset of B" << endl;
```

```
    if (IsProperSubset(A, B))
        cout << "\nA is a proper subset of B" << endl;
    else
        cout << "\nA is not a proper subset of B" << endl;

    //Of my own design.

    cout << "\nCardinality (A union complement (B difference C)): ";
    D = Union(A, Complement(Difference(B, C)));
    cout << Cardinality(D);
    cout << endl << endl;

    if (IsSubset(Intersection(A, B), A))
        cout << "Intersection (A, B) is a subset of A" << endl;
    else
        cout << "Intersection (A, B) is not a subset of A" << endl;
}
```

## Output:

The following outputs use the same testing code. However, the sets change in each.

A = {Cat, Dog, Fish}
B = {Cat, Dog, Liger}
C = {Dog, Liger, Snake Turtle}

```
C:\WINDOWS\system32\cmd.exe
Set A: { Cat, Dog, Fish }
Set B: { Cat, Dog, Liger }
Set C: { Dog, Liger, Snake, Turtle }

(A union B) Intersection C: { Dog, Liger }

A union (B intersection C): { Cat, Dog, Fish, Liger }

Complement (A intersection B): { Bat, Chimp, Fish, Liger, Snake, Turtle }

Complement A union complement B: { Bat, Chimp, Fish, Liger, Snake, Turtle }

A difference B: { Fish }

Powerset(A):
{  }
{ Fish }
{ Dog }
{ Dog, Fish }
{ Cat }
{ Cat, Fish }
{ Cat, Dog }
{ Cat, Dog, Fish }

A is not a subset of B

A is not a proper subset of B

Cardinality (A union complement (B difference C)): 8

Intersection (A, B) is a subset of A
Press any key to continue . . .
```

A = {Bat, Chimp, Liger, Snake, Turtle}

B = {Bat, Cat, Chimp, Dog, Fish}

C = {Dog, Fish, Liger, Snake Turtle}

```
C:\WINDOWS\system32\cmd.exe
Set A: { Bat, Chimp, Liger, Snake, Turtle }
Set B: { Bat, Cat, Chimp, Dog, Fish }
Set C: { Dog, Fish, Liger, Snake, Turtle }

(A union B) Intersection C: { Dog, Fish, Liger, Snake, Turtle }

A union (B intersection C): { Bat, Chimp, Dog, Fish, Liger, Snake, Turtle }

Complement (A intersection B): { Cat, Dog, Fish, Liger, Snake, Turtle }

Complement A union complement B: { Cat, Dog, Fish, Liger, Snake, Turtle }

A difference B: { Liger, Snake, Turtle }

Powerset(A):
{  }
{ Turtle }
{ Snake }
{ Snake, Turtle }
{ Liger }
{ Liger, Turtle }
{ Liger, Snake }
{ Liger, Snake, Turtle }
{ Chimp }
{ Chimp, Turtle }
{ Chimp, Snake }
{ Chimp, Snake, Turtle }
{ Chimp, Liger }
{ Chimp, Liger, Turtle }
{ Chimp, Liger, Snake }
{ Chimp, Liger, Snake, Turtle }
{ Bat }
{ Bat, Turtle }
{ Bat, Snake }
{ Bat, Snake, Turtle }
{ Bat, Liger }
{ Bat, Liger, Turtle }
{ Bat, Liger, Snake }
{ Bat, Liger, Snake, Turtle }
{ Bat, Chimp }
{ Bat, Chimp, Turtle }
{ Bat, Chimp, Snake }
{ Bat, Chimp, Snake, Turtle }
{ Bat, Chimp, Liger }
{ Bat, Chimp, Liger, Turtle }
{ Bat, Chimp, Liger, Snake }
{ Bat, Chimp, Liger, Snake, Turtle }

A is not a subset of B

A is not a proper subset of B

Cardinality (A union complement (B difference C)): 7

Intersection (A, B) is a subset of A
Press any key to continue . . .
```

A = {Bat, Dragon, Hippopotamus, Gecko, Sloth, Deer, Kangaroo}

B = {Hippopotamus, Gecko, Sloth, Bat, Rhinoceros, Squirrel, Platypus}

C = {Gecko, Sloth, Bat, Rhinoceros, Dog, Fish, Horse, Snake, Turtle, Donkey, Gorilla, Llama}

```
Select C:\WINDOWS\system32\cmd.exe
Set A: { Bat, Dragon, Hippopotamus, Gecko, Sloth, Deer, Kangaroo }
Set B: { Bat, Hippopotamus, Rhinoceros, Gecko, Sloth, Squirrel, Platypus }
Set C: { Bat, Dog, Fish, Snake, Turtle, Horse, Donkey, Llama, Rhinoceros, Gecko, Sloth, Gorilla }

(A union B) Intersection C: { Bat, Rhinoceros, Gecko, Sloth }

A union (B intersection C): { Bat, Dragon, Hippopotamus, Rhinoceros, Gecko, Sloth, Deer, Kangaroo }

Complement (A intersection B): { Cat, Chimp, Dog, Fish, Liger, Snake, Turtle, Bear, Dragon, Horse, Wolf, Rat, Gerbil,
 Rabbit, Monkey, Donkey, Llama, Zebra, Rhinoceros, Frog, Deer, Kangaroo, Gorilla, Alligator, Panda, Squirrel, Duck, P
latypus }

Complement A union complement B: { Cat, Chimp, Dog, Fish, Liger, Snake, Turtle, Bear, Dragon, Horse, Wolf, Rat, Gerbi
l, Rabbit, Monkey, Donkey, Llama, Zebra, Rhinoceros, Frog, Deer, Kangaroo, Gorilla, Alligator, Panda, Squirrel, Duck,
 Platypus }

A difference B: { Dragon, Deer, Kangaroo }

Powerset(A):
{  }
{ Kangaroo }
{ Deer }
{ Deer, Kangaroo }
{ Sloth }
{ Sloth, Kangaroo }
{ Sloth, Deer }
{ Sloth, Deer, Kangaroo }
{ Gecko }
{ Gecko, Kangaroo }
{ Gecko, Deer }
{ Gecko, Deer, Kangaroo }
{ Gecko, Sloth }
{ Gecko, Sloth, Kangaroo }
{ Gecko, Sloth, Deer }
{ Gecko, Sloth, Deer, Kangaroo }
{ Hippopotamus }
{ Hippopotamus, Kangaroo }
{ Hippopotamus, Deer }
{ Hippopotamus, Deer, Kangaroo }
{ Hippopotamus, Sloth }
{ Hippopotamus, Sloth, Kangaroo }
{ Hippopotamus, Sloth, Deer }
{ Hippopotamus, Sloth, Deer, Kangaroo }
{ Hippopotamus, Gecko }
{ Hippopotamus, Gecko, Kangaroo }
{ Hippopotamus, Gecko, Deer }
{ Hippopotamus, Gecko, Deer, Kangaroo }
{ Hippopotamus, Gecko, Sloth }
{ Hippopotamus, Gecko, Sloth, Kangaroo }
{ Hippopotamus, Gecko, Sloth, Deer }
{ Hippopotamus, Gecko, Sloth, Deer, Kangaroo }
{ Dragon }
{ Dragon, Kangaroo }
{ Dragon, Deer }
{ Dragon, Deer, Kangaroo }
{ Dragon, Sloth }
{ Dragon, Sloth, Kangaroo }
{ Dragon, Sloth, Deer }
{ Dragon, Sloth, Deer, Kangaroo }
{ Dragon, Gecko }
{ Dragon, Gecko, Kangaroo }
{ Dragon, Gecko, Deer }
{ Dragon, Gecko, Deer, Kangaroo }
```

(part 1 of 3)

```
Select C:\WINDOWS\system32\cmd.exe
{ Dragon, Gecko, Sloth }
{ Dragon, Gecko, Sloth, Kangaroo }
{ Dragon, Gecko, Sloth, Deer }
{ Dragon, Gecko, Sloth, Deer, Kangaroo }
{ Dragon, Hippopotamus }
{ Dragon, Hippopotamus, Kangaroo }
{ Dragon, Hippopotamus, Deer }
{ Dragon, Hippopotamus, Deer, Kangaroo }
{ Dragon, Hippopotamus, Sloth }
{ Dragon, Hippopotamus, Sloth, Kangaroo }
{ Dragon, Hippopotamus, Sloth, Deer }
{ Dragon, Hippopotamus, Sloth, Deer, Kangaroo }
{ Dragon, Hippopotamus, Gecko }
{ Dragon, Hippopotamus, Gecko, Kangaroo }
{ Dragon, Hippopotamus, Gecko, Deer }
{ Dragon, Hippopotamus, Gecko, Deer, Kangaroo }
{ Dragon, Hippopotamus, Gecko, Sloth }
{ Dragon, Hippopotamus, Gecko, Sloth, Kangaroo }
{ Dragon, Hippopotamus, Gecko, Sloth, Deer }
{ Dragon, Hippopotamus, Gecko, Sloth, Deer, Kangaroo }
{ Bat }
{ Bat, Kangaroo }
{ Bat, Deer }
{ Bat, Deer, Kangaroo }
{ Bat, Sloth }
{ Bat, Sloth, Kangaroo }
{ Bat, Sloth, Deer }
{ Bat, Sloth, Deer, Kangaroo }
{ Bat, Gecko }
{ Bat, Gecko, Kangaroo }
{ Bat, Gecko, Deer }
{ Bat, Gecko, Deer, Kangaroo }
{ Bat, Gecko, Sloth }
{ Bat, Gecko, Sloth, Kangaroo }
{ Bat, Gecko, Sloth, Deer }
{ Bat, Gecko, Sloth, Deer, Kangaroo }
{ Bat, Hippopotamus }
{ Bat, Hippopotamus, Kangaroo }
{ Bat, Hippopotamus, Deer }
{ Bat, Hippopotamus, Deer, Kangaroo }
{ Bat, Hippopotamus, Sloth }
{ Bat, Hippopotamus, Sloth, Kangaroo }
{ Bat, Hippopotamus, Sloth, Deer }
{ Bat, Hippopotamus, Sloth, Deer, Kangaroo }
{ Bat, Hippopotamus, Gecko }
{ Bat, Hippopotamus, Gecko, Kangaroo }
{ Bat, Hippopotamus, Gecko, Deer }
{ Bat, Hippopotamus, Gecko, Deer, Kangaroo }
{ Bat, Hippopotamus, Gecko, Sloth }
{ Bat, Hippopotamus, Gecko, Sloth, Kangaroo }
{ Bat, Hippopotamus, Gecko, Sloth, Deer }
{ Bat, Hippopotamus, Gecko, Sloth, Deer, Kangaroo }
{ Bat, Dragon }
{ Bat, Dragon, Kangaroo }
{ Bat, Dragon, Deer }
{ Bat, Dragon, Deer, Kangaroo }
{ Bat, Dragon, Sloth }
{ Bat, Dragon, Sloth, Kangaroo }
{ Bat, Dragon, Sloth, Deer }
{ Bat, Dragon, Sloth, Deer, Kangaroo }
{ Bat, Dragon, Gecko }
{ Bat, Dragon, Gecko, Kangaroo }
{ Bat, Dragon, Gecko, Deer }
```

(part 2 of 3)

```
Select C:\WINDOWS\system32\cmd.exe
{ Bat, Dragon, Gecko, Deer, Kangaroo }
{ Bat, Dragon, Gecko, Sloth }
{ Bat, Dragon, Gecko, Sloth, Kangaroo }
{ Bat, Dragon, Gecko, Sloth, Deer }
{ Bat, Dragon, Gecko, Sloth, Deer, Kangaroo }
{ Bat, Dragon, Hippopotamus }
{ Bat, Dragon, Hippopotamus, Kangaroo }
{ Bat, Dragon, Hippopotamus, Deer }
{ Bat, Dragon, Hippopotamus, Deer, Kangaroo }
{ Bat, Dragon, Hippopotamus, Sloth }
{ Bat, Dragon, Hippopotamus, Sloth, Kangaroo }
{ Bat, Dragon, Hippopotamus, Sloth, Deer }
{ Bat, Dragon, Hippopotamus, Sloth, Deer, Kangaroo }
{ Bat, Dragon, Hippopotamus, Gecko }
{ Bat, Dragon, Hippopotamus, Gecko, Kangaroo }
{ Bat, Dragon, Hippopotamus, Gecko, Deer }
{ Bat, Dragon, Hippopotamus, Gecko, Deer, Kangaroo }
{ Bat, Dragon, Hippopotamus, Gecko, Sloth }
{ Bat, Dragon, Hippopotamus, Gecko, Sloth, Kangaroo }
{ Bat, Dragon, Hippopotamus, Gecko, Sloth, Deer }
{ Bat, Dragon, Hippopotamus, Gecko, Sloth, Deer, Kangaroo }

A is not a subset of B

A is not a proper subset of B

Cardinality (A union complement (B difference C)): 30

Intersection (A, B) is a subset of A
Press any key to continue . . .
```

(part 3 of 3)

## Conclusion

This assignment was much easier than I thought. The implementation of several of these functions was literally one line of code! However, the powerSet function did give me a little trouble. Yet, despite this, I managed to complete everything in record time. Looking forward to the next assignment!