CISP 440

Austin Smothers

Homework 4

/\*

Set operations

Austin Smothers

February 8, 2017

CISP 440

Professor Ross

Performs set operations.

Universe = {Bat, Cat, Chimp, Dog, Fish, Liger, Snake, Turtle}

This code builds on code provided by Professor Ross

\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#pragma warning( disable : 4996)

#pragma warning( disable : 4244)

// Start with a small universe

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

typedef unsigned char set; // a set, by any other name, would smell as sweet.

// Then use this big 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 long int set32; // a set, but bigger

/\*

Prints out a set in set-sequence notation

\*/

void printSet(set32 A)

{

printf("{ ");

bool commaflag = false;

int i = 0;

unsigned char mask = 0x80;

for( ; mask; mask >>= 1, i++) {

if(mask & A)

{

if(commaflag) printf(", ");

printf(BigUniverse[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(set32 & A, char str[])

{

// get a unique hash for each string

int hash = (str[0] + str[2]) % 20;

// map unique string hashes back to their Universe indexes

// 0 1 2 3 4 5 6 7 8 9

int g[20] = { 6, -1, 0, 1, -1, 4, -1, -1, -1, -1,

// 10 11 12 13 14 15 16 17 18 19

-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;

}

////////////// I HAVE DONE A MIXTURE OF WRITING AND COPY/PASTING THE FOLLOWING /////////////////

/\*

Union

\*/

set32 Union(set32 A, set32 B)

{

//this line adds B to A

A = A | B;

return A;

}

/\*

Intersection

\*/

set32 Intersection(set32 A, set32 B)

{

//this function sets A = the common elements of A and B

A = A & B;

return A;

}

/\*

Complement

\*/

set32 Complement(set32 A)

{

//flips A to contain the entire universe that it didn’t contain

A = ~A;

return A;

}

/\*

Difference

\*/

set32 Difference(set32 A, set32 B)

{

//this function sets A = The elements of A not present in B

A = Intersection(A, Complement(B));

return A;

}

/\*

Cardinality

\*/

int Cardinality(set32 A)

{

int b = 0;

/\*this set of code was taken from the printSet function

becuase it already functionally manipulates sets

and getting C++ to function with such functions as A.size()

was proving too much of a pain in the butt\*/

//this check for masking is likely redundant, but

//due to my lack of familiarity with masking, I prefer

//to play it safe

int i = 0;

unsigned char mask = 0x80;

for (; mask; mask >>= 1, i++) {

if (mask & A)

{

b++; //b functions as a counter which increments for each

//member of the set

}

}

return b;

//easy enough, just return the size of the set to get its cardinality

}

void printPowerSet(set32 A)

{

//no clue how to do this part. Oh well.

}

bool IsSubset(set32 ASubset, set32 ASet)

{

set32 C = Difference(ASubset, ASet);

//uses the difference function to determine

//if ASubset has any elements which aren't in ASet

set32 D = 0;

//if ASubset has any such elements, ASubset

//isn't a subset of ASet

if (C != D)

return false;

else

return true;

}

bool IsProperSubset(set32 ASubset, set32 ASet)

{

if (ASubset == ASet) //tests if the two sets are the same

return false; //If they are the same, they aren't proper subsets

//Now that we've established whether or not the two sets are the same,

//we can use IsSubset to test if ASubset is a subset of ASet

//If the prior statement is true, ASubset is a proper subset of ASet

else

{

bool Tf = IsSubset(ASubset, ASet);

return Tf;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

main - this be where the program starts...or...is it?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

set32 A = 0;

insert(A, "Bat");

insert(A, "Cat");

insert(A, "Chimp");

insert(A, "Snake");

printf("Set A: ");

printSet(A);

printf("\nCardinality: ");

printf("%d", Cardinality(A));

printf("\n\nPowerSet(A):\n");

printPowerSet(A);

set32 B = 0;

insert(B, "Chimp");

insert(B, "Fish");

insert(B, "Liger");

printf("\nSet B: ");

printSet(B);

set32 C = 0;

insert(C, "Chimp");

insert(C, "Fish");

insert(C, "Liger");

printf("\nSet C: ");

printSet(C);

printf("\n(A U B) n ~C: ");

set32 D = Intersection(Union(A, B), ~C);

printSet(D);

//Added a statement to print the Complement of A

printf("\nA': ");

set32 E = Complement(A);

printSet(E);

//Added a statement to print the Difference of A and B

printf("\nA n B: ");

set32 F = Difference(A, B);

printSet(F);

/\*

Operations Lightning round

\*/

//(A Union B) Intersection C

printf("\n(A U B) n C: ");

F = Intersection(Union(A, B), C);

printSet(F);

//A Union (B Intersection C)

printf("\nA U (B n C): ");

F = Union(A, Intersection(B, C));

printSet(F);

//~(A Intersection B)

printf("\n~(A n B): ");

F = Complement(Intersection(A, B));

printSet(F);

//A Difference B

printf("\nA - B: ");

F = Difference(A, B);

printSet(F);

//~(A Difference B)

printf("\n~(A - B): ");

F = Complement(Difference(A, B));

printSet(F);

//(A Union B) Difference C

printf("\n(A U B) - C: ");

F = Difference(Union(A, B), C);

printSet(F);

if(IsSubset(B, C))

printf("\nB is a subset of C");

else

printf("\nB is NOT a subset of C");

if(IsProperSubset(B, C))

printf("\nB is a proper subset of C");

else

printf("\nB is NOT a proper subset of C");

printf("\n");



