SCC111: GOING 'SOCIAL'

THE GOAL

This week's lab aims to:

- Practice working with arrays in C (again)
- This time choosing how to represent and solve problems relating to sets and functions!
- You'll also get to practice your 'computational problem solving', working from a high level problem specification to a C solution

PROBLEM STATEMENT

Social media is 'a thing'… :) 'The Graph' of relations between people is sometimes seen as a powerful expression of affinity of interests, predictive of voting patterns, attitudes, relationships. It's the 'grist to the mill' of the advertising machines that drive billion dollar businesses.

The goal of the problem today is to represent and explore relationships between a (small) set of people.

To make this a little more tractable at this stage of the course, we would suggest representing a social media account as an integer, and a 'binary relation' (link/ friend relationship) as a tuple.

Thus,

> (1,2)

represents that `2` is a friend of `1`, where `1` and `2` are social media account holders.

Your goal is to represent a small graph of relations as a set (we've given you a basic set implementation to help, see below).

You should then implement the following functions to test and examine the relationships between your users.

1. Firstly, mental health is important - especially on social media, who is a friend to themselves? (i.e. are there any reflexive relationships?)

2. Secondly, one of the ironies of social media - you can be a friend with another account, without someone being a friend back (asymmetry). Can you identify all the symmetric 'or reciprocal' friendships?

SUPPORTING THE TASK

To make the task a little easier to get traction on, we are providing a set of functions that allow you to use an 'int array' as a 'set' of tuples or pairs, such as the above.

You can call the following functions:

```
void init_set(int set[]);  // Initialise the array 'set' (must be at least 20 long!)
int in_set(int set[], int a, int b); // Test if tuple (a,b) is in 'set'
int add_to_set(int set[], int a, int b); // Add (a,b) to the set
int remove_from_set(int set[], int a, int b); // Remove (a,b) from the set
int num_elements(int set[]); // How many tuples in the set
void print_set(int set[]); // Print out the set using printf
int get_tuple_at(int set[], int index, int *a, int *b) // Get tuple at position (from
↪  0...)
```

Each set element (tuple with a pair of integers) takes up 2 spaces in the array, so if you want to store 10 tuples, you'll need an array of length 20. *The predefined constant* `MAX_SET` *is defined as 20.*

Note also that ' `get_tuple_at` ' expects the tuple index number (from 0), not the array index.

Example use:

```
// Declare and initialise the set
int set[MAX_SET];
init_set(set);

// Add some tuples
add_to_set(set, 1, 1);
add_to_set(set, 2, 2);
```

```
printf("in_set(1,1) = %d\n", in_set(set, 1, 1));
```

The tuples are stored as consecutive values in the array terminated by the special value `-1`, so if `(a,b)` is the tuple, `set[0]` is value `a` and `set[1]` is value `b`.

See the following for an example of scanning the set looking at the values:

```
void print_set(int set[])
{
    for (int i = 0; i < MAX_SET && set[i] >= 0; i += 2)
    {
        printf("(%d,%d)\n", set[i], set[i + 1]);
    }

    printf("\n");
}
```

To compile your program, you'll need to download the project zipfile containing the files `miniset.c` and `miniset.h` files and compile your project like this (assuming your source file is called `social.c`):

```
gcc -o social social.c miniset.c
```

Alternatively, you can also use the provided Makefile to simplify the built process by typing the command `make` from your terminal while you are in the project folder (Makefile is a built automation tool and you can search online to find out more).

We also provide an initial version of `social.c` as a starting point. This already has the header files included you'll need for this project. You are expected to implement the functions: `int is_reflexive(int set[])` and `int is_symmetric(int set[])`, as well as, add code in the main function that initializes a set and runs the two functions to test for properties.

For example, if you use the graph of the following function:

Your main method could include code that generates the following output:

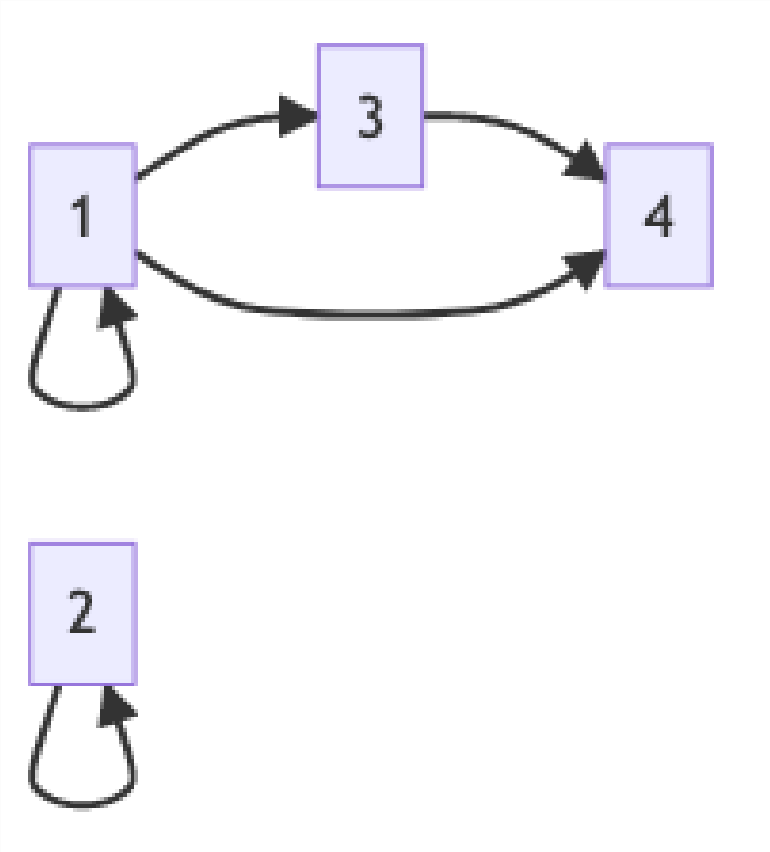

Figure 1:

```
print_set: (1,1), (2,2), (1,3), (3,4), (1,4)
in_set(0,0) = 0
in_set(1,1) = 1
is_reflexive() = 0
is_symmetric() = 0
is_transitive() = 1
```

SCC.121: PROPOSITIONAL LOGIC

This week's lab activities will cover the topic of *Propositional Logic*. Propositional logic is a fundamental aspect of computer science due to its role in establishing the foundation for structured thinking and problem-solving. It provides a system for formulating and evaluating statements in a binary format, where each statement is either true or false. This binary nature aligns perfectly with the basic principles of computer operations, which are fundamentally based on binary code (0s and 1s). Propositional logic allows you to create algorithms and design circuits that perform logical operations, crucial for tasks ranging from basic computing processes to complex decision-making in artificial intelligence. Moreover, propositional logic is essential in the development of software, where logical consistency and error-free operations are paramount.

THE GOAL

Using the knowledge of the lecture material from Week 5, you are expected to work through the following questions. These problems will equip you with the essential knowledge and skills to analyze express statements and prove their correctness. You should work out the answers *with a pen and paper*!

1. Let W = "it is weekend", S = "swimming is fun", C = "coursework is done", and L = "Logic is easy"

   - Translate the following proposition into the most natural equivalent statement in English. Try to make the sentence as simple and as natural as possible:

   $$(L \rightarrow C) \wedge (W \rightarrow S)$$

   - Translate the following statement into propositional logic: "Logic is easy or swimming is fun, if it is weekend and the coursework is done".

2. Let P = "It is raining"
   Let Q = "Mary is sick"
   Let T= "Bob stayed up late last night"
   Let R = "Paris is the capital of France" Let S= "John is angry"
   Given the above atomic propositions, translate the following ones into propositional logic:

- It isn't raining
- Bob stayed up late last night and John is angry
- Mary is sick or Mary isn't sick
- It is not the case that if it is raining then John isn't angry
- Bob did not stay up late last night

3. Given the atomic propositions described in Exercise 2, translate the following ones into propositional logic:

   - It is not the case that Mary is sick or Bob stayed up late last night
   - John is angry but Mary isn't sick
   - Mary is sick and it is raining implies that Bob stayed up late last night
   - John is in no way angry

4. Given the following propositions:
   C = "The cheesecake is good."
   F = "The fries are greasy."
   W = "The wings are spicy."
   Translate the following logical statements into words:

   - $(\sim C \wedge F) \rightarrow W$
   - $\sim (C \vee W)$
   - $\sim (\sim W \wedge C)$
   - $\sim (\sim F)$

5. Consider the fundamental connectives OR and XOR.

   - Complete their truth tables.
   - Explain how they differ.
   - Provide an example of statements and how they are connected through each of these logical operators by writing the two result propositions in both predicate logic language and English.

6. Consider the fundamental connectives conditional and biconditional.

   - Complete their truth tables.

- Explain how they differ.
- Provide an example of statements and how they are connected through each of these logical operators by writing the two result propositions in both predicate logic language and English.

7. Use the truth tables to prove them the following equivalence. They are called annihilation laws:

   - $P \wedge F \equiv F$
   - $P \vee T \equiv T$

8. This is one variant of the absorption law whose proof has not been shown in the class: the conjunction of any proposition P with $(P \vee Q)$ has the same truth value as P.
   $P \wedge (P \vee Q)$ is equivalent to P Prove this equivalence by using truth table.

9. Prove the equivalence from the Exercise 2 by applying transformation rules. *Hint: use distribution, identity, and annihilation laws.*

   - distributive law: $P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$
   - identity law: $P \wedge T \Leftrightarrow P$
   - annihilation law: $P \vee T \leftrightarrow T$
   - idempotence law: $P \wedge P \leftrightarrow P$

10. Is the following proposition a tautology?
    $(A \vee B) \vee (\ A \wedge\ B)$
    To answer this, use:

- transformation rules
- truth table

SCC.121: PREDICATE LOGIC

1. Which ones of the following are formulae, and if so are they open, closed, or quantified?

   - x = y - 2
   - x
   - Joe
   - swims
   - Joe swims
   - Every person swims

2. For the formula in Exercise1(a), what are the truth values for:

   - P(1, 1)
   - P(0, 2)
   - P(2, 4)
   - P(0, 0)

3. Let A(c, n) = "Computer c is connected to network n", where c is a variable representing computers and n is a variable representing networks.
   Suppose that the computer MATH1 is connected to network CAMPUS2, but not to network CAMPUS1. What are the truth values of:

   - A(m1, c1)
   - A(m1, c2)?

4. Consider the following atomic formulae:

   - "Adam is tall"
   - "Beth is tall"
   - "Carl is tall"
   - "Zeke is tall"

How can we capture them all in a single atomic formula?

5. Let $P(x)$ be "2x is even". In which Universe of Discourse is P(x) TRUE?

   - The set of real numbers
   - The set of integers

6. What is the negation of the formula:

   - "All British people eat fish and chips"
   - "No British people eat fish and chips"
   - "There is a British person who does not eat fish and chips"
   - "All British people eat no fish and chips"

7. Let R(x,y) be "x beats y" in rock scissors paper game.

Rock scissors paper game:

- rock smashes scissors,
- scissors cuts paper,
- paper covers rock.

What is the truth values for:

1. R(paper, rock)
2. R(paper, scissors)
3. R(scissors, rock)

SCC.131: LOGIC CIRCUIT

In this week's lab activities, we will practise applying and combining truth table, Karnaugh maps and de Morgan's Law to design different logic circuits such as a two-out-of-three voter logic circuit and a half adder.

THE GOAL

By the end of this task you should be able to:

- ☐ Be familiar with designing a logic circuit of a two-out-of-three voter using NAND gates;
- ☐ Be familiar with designing a logic circuit of a half adder using NAND gates.

TODAY'S TASK

This is a worksheet for the SCC.131 week 6 lab practical. Work through the following questions and ask us if you need help. You should work out the answers *with a pen and paper* - no calculators nor computers!

1. Design a two-out-of-three voter logic circuit using NAND gates only:

- The circuit has 3 inputs (A, B and C) and 1 output (see Figure 1)
- The output is 1 *if two or more of the inputs are 1*

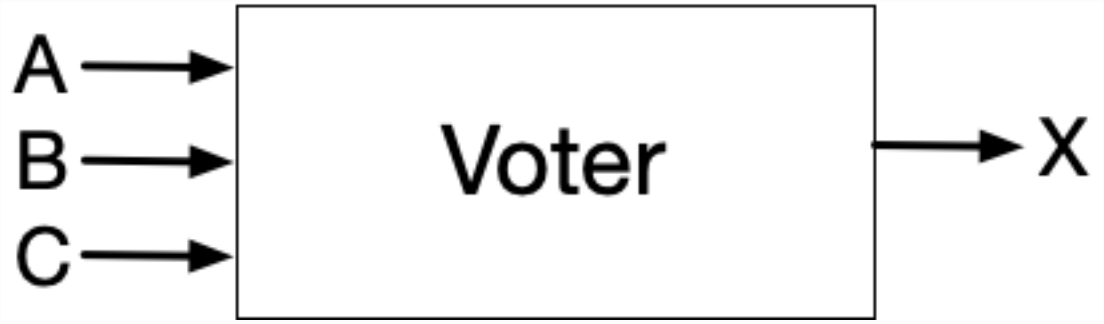

Figure 2:

This type of circuit is often used as a fail-safe device in computer systems that require a very high degree of reliability. In such systems, all functions are "triple-redundant", and voter circuits are used to arbitrate the

three outputs. The voters eliminate the output from any single one of the triple-redundant functions that disagrees with the other two.

How to proceed:

- Draw a truth table with 3 input columns and 1 output column.
- Map the resulting Boolean expression, in sum-of-products form, onto a 3-input Karnaugh map.
- Use this to derive a minimised Boolean expression (remember to consider wrap-around!).
- Prove by perfect induction (i.e. by extending your truth table) that your minimised expression is indeed equivalent to your initial sum-of-products form.
- Implement your minimised expression using NAND gates only (apply de Morgan's Law and draw the circuit).

2. Design a half adder logic circuit using NAND gates only.

How to proceed:

- A half adder has two inputs, A and B, and two outputs, SUM and CARRY.
- Draw the truth tables for both the SUM and CARRY logical functions.
- Derive the resulting Boolean expressions for both the SUM and CARRY, in sum-of-products form.
- Implement your expressions using NAND gates only (apply de Morgan's Law and draw the circuit).

HACKER EDITION

SCC.111: GOING 'SOCIAL'

Advertisers definitely want to understand the graph a little better. Can you write a function e.g. `is_transitive(int set[])` to identify whether the relationships in the set are transitive, i.e. if `(1,2)` and `(2,3)` are friends, are `(1,3)` also friends?

Hint: you may find it helpful to build a matrix mapping the 'binary relations' between all the pairs of accounts. For example, you can declare a two dimensional array map like this:

```
int map[10][10];
```

A 10 x 10 2D array of integers. And access each cell with two subscripts as follows:

```
printf("Map value at (1,2) is:\n", map[1][2]);
```

Don't forget to show us your solutions!

SCC121: PROPOSITIONAL LOGIC

1. Is the following proposition a tautology?

$$(A \vee B) \vee (\ A \wedge\ B)$$

   To answer this, use:

   - transformation rules
   - truth table

2. Let this be an argument:

   If Mike gambles at casino, then Monica will be angry.

   If John plays cards at casino, then Jess will be angry.

   If Monica or Jess gets angry, then Angie (their solicitor) will be notified.

   Angie has not heard from either of these two clients.

Consequently, Mike didn't gamble at casino and John didn't play cards at casino.

- Write the argument in propositional logic
- Proof its validity

3. Prove the hypothetical syllogism using inference and/or transformation rules.

4. Fill in the following arguments with the right conclusions so that they become valid inferences:

   If there are more passengers than there are seats in the airplane,
   then at least one passenger has to take a later flight.
   There are more passengers than there are seats.
   ∴

   If 19 is divisible by 6, then it is divisible by 3.
   19 is not divisible by 3.
   ∴

5. Show that the premises:

   - It is not sunny this afternoon and it is colder than yesterday.
   - We will go swimming only if it is sunny.
   - If we do not go swimming, then we will take a canoe trip.
   - If we take a canoe trip, then we will be home by sunset.

   lead to the conclusion:

   - We will be home by the sunset.

   Write a formal proof, a sequence of steps that state premises or apply inference rules to previous steps.
   *Note: you should translate all atomic propositions into propositional logic.

SCC.121: PREDICATE LOGIC

1. Let P(x, y) be: "x + y > 4", where the Universe of discourse is all integer pairs. Which of the following are open or closed formulae, and if closed, which is the truth value?

- P(1, 2)
- P(1000, 2)
- P(1000, y)
- P(x, y)

2. Transform the following English sentences into formulae of predicate logic:

   - "All cats are mammals"
   - "There is a black cat"

3. Let: L(x,y) = "x loves y" on the Universe of people. Translate:

   - Everybody loves Raymond.
   - Everybody loves somebody.
   - There is somebody whom everybody loves.
   - There is somebody who Raymond doesn't love.
   - There is somebody whom no one loves.

4. Let the Universe of discourse be: Homer, Marge, Bart, Lisa, Maggie.
   Let the formula be:

Brother (x, y)	"x is y's brother"
Sister (x, y)	"x is y's sister"
Mother (x, y)	"x is y's mum"
Father (x, y)	"x is y's dad"

Father (Homer, Bart), Father (Homer, Lisa), Father (Homer, Maggie)	True
Mother (Marge, Bart), Mother (Marge, Lisa), Mother (Marge, Maggie)	True

Sister (Lisa, Bart), Sister (Lisa, Maggie), Sister (Maggie, Bart), Sister (Maggie, Lisa)	True
Brother (Bart Lisa), Brother (Bart, Marge)	True

Translate in predicate logic:

- Marge is Lisa's mother but she is not Homer's mother.
- All of Homer's kids are also Marge's kids.
- There is a kid whose father is Homer and whose mother is Marge.

SCC.131: LOGIC CIRCUIT

1. Simplify your half adder logic circuit using 5 NAND gates only. You need to apply Boolean algebraic manipulations to simplify your half adder logic circuit design.