# ENS1161 Assignment 1

## Question 1

### A: Consider the following argument

If bears are brown then giraffes are not green. If bears are not brown then rabbits are red. Rabbits are red. Therefore giraffes are not green.

* b = bears are brown
* g = giraffes are green
* r = rabbits are red

1. State the argument in symbols
   * b → ~g, ~b → r, r | ~g
2. Rewrite the symbolic argument as a proposition
   * (~b ˄ r) → ~g
3. Construct a truth table for the proposition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| b | r | g | ~b ˄ r | ~g | (~b ˄ r) → ~g |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

1. State whether the argument is valid.
   * Proposition is equal to argument, therefore argument is valid

### B: The propositions below can be arranged in three groups so that each member of a group is logically equivalent to the other two. Find the three groups.

1. Either the light is not on or the system is ready.
   * ~l ˅ s
2. If the system is ready then the light is on.
   * s → l
3. If the light is on then the system is not ready.
   * l → ~s
4. If the light is not on then the system is not ready.
   * ~l → ~s
5. If the system is not ready then the light is not on.
   * ~s → ~l
6. Either the system is not ready or the light is on.
   * ~s ˅ l
7. If the light is on then the system is ready.
   * l → s
8. If the system is ready then the light is not on.
   * s → ~l
9. Either the system is not ready or the light is not on.
   * ~s ˅ ~l

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **s** | **l** | **~s** | **~l** | **~l ˅ s** | **s → l** | **l → ~s** | **~l → ~s** | **~s → ~l** | **~s ˅ l** | **l → s** | **s → ~l** | **~s ˅ ~l** |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

#### Group 1:

|  |  |
| --- | --- |
| **Proposition Number** | **Symbolic** |
| 1 | ~l ˅ s |
| 5 | ~s → ~l |
| 7 | l → s |

#### Group 2:

|  |  |
| --- | --- |
| **Proposition Number** | **Symbolic** |
| 2 | s → l |
| 4 | ~l → ~s |
| 6 | ~s ˅ l |

#### Group 3:

|  |  |
| --- | --- |
| **Proposition Number** | **Symbolic** |
| 3 | l → ~s |
| 8 | s → ~l |
| 9 | ~s ˅ ~l |

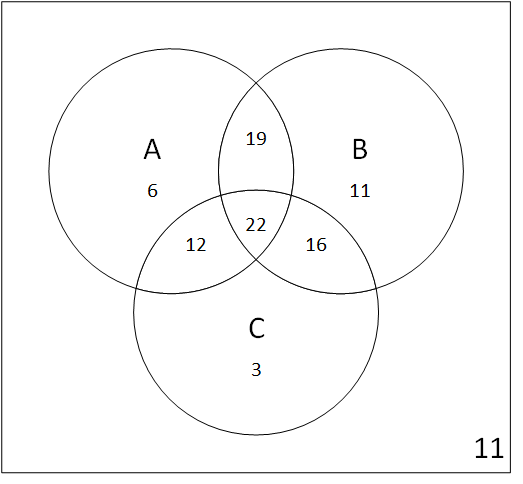
## 

## Question 2

A group of 100 students is polled to see how many watched three TV shows, Action, Buzz, and Calypso. The results showed that:

* 59 watched Action
  + denoted by A
* 68 watched Buzz
  + denoted by B
* 52 watched Calypso
  + denoted by C
* 41 watched Action and Buzz
* 34 watched Action and Calypso
* 38 watched Buzz and Calypso
* 11 did not watch any of the three.

### A: Calculate the number of students in each of the eight subsets shown in the Venn diagram. Copy the Venn diagram and enter the number of students in each subset.



venn movies

### B: Hence find how many students watched

1. Action and Calypso, but not Buzz
   * 12
2. Buzz only
   * 11
3. Only two of the three shows
   * 47
4. At least two of the shows
   * 69

## Question 3

Suppose that P is a set of people and M is a set of movies. Define the predicate S(p, m) to mean “person p has seen movie m”. Consider the sentences below. For each sentence, the negation is also in the list. Match each sentence with its negation. Give your answer by listing the number of each sentence and its negation, for example “4 and 10”.

1. There is at least one movie that some person has seen
   * ∃m ∈ M, ∃p ∈ P, S(p, m)
2. For each movie there is somebody who has not seen it
   * ∀m ∈ M, ∃p ∈ P, ~S(p, m)
3. There is some movie that nobody has seen
   * ∃m ∈ M, ∀p ∈ P, ~S(p, m)
4. There is at least one movie that everybody has seen
   * ∃m ∈ M, ∀p ∈ P, S(p, m)
5. For each movie there is somebody who has seen it
   * ∀m ∈ M, ∃p ∈ P, S(p, m)
6. There is some person who has not seen any of the movies
   * ∃p ∈ P, ∀m ∈ M, ~S(p, m)
7. There is at least one movie that some person has not seen
   * ∃m ∈ M, ∃p ∈ P, ~S(p, m)
8. Nobody has seen any of the movies
   * ∀m ∈ M, ∀p ∈ P, ~S(p, m)
9. For each person there is some movie that they have not seen
   * ∃p ∈ P, ∃m ∈ M, ~S(p, m)
10. Everybody has seen at least one movie
    * ∀p ∈ P, ∃m ∈ M, S(p, m)
11. There is some person who has seen all the movies
    * ∃p ∈ P, ∀m ∈ M, S(p, m)
12. Everybody has seen all of the movies
    * ∀m ∈ M, ∀p ∈ P, S(p, m)

### Answer

1. 1 and 8
2. 2 and 4
3. 3 and 5
4. 6 and 10
5. 7 and 12
6. 9 and 11

## Question 4

### Introduction:

There is a simple way of representing sets using so-called *bit strings*. A bit string is simply a string of 0’s and 1’s. Suppose that U is the universal set. Then the rule for the bit string for a particular set S is:

If the kth element of U is in S, then the kth bit of the bit string is 1. If the kth element of U is not in S, then the kth bit of the bit string is 0.

### Example 1:

* Suppose the universal set is U = {1, 2, 3, 4, ..., 10}
* Then set A = {1, 3, 5, 9} is represented by bit string 1010100010
  + The 1st element of U appears in set A, so the 1st element of the bit string is 1
  + The 2nd element of U does not appear in set A, so the 2nd element of the bit string is 0
* The set {5, 6, 7} is represented by bit string 0000111000
* Conversley, the bit string 0110010001 represents the set C = {2, 3, 6, 10}
  + The 1st element of the bit string is 0, so the 1st element of U (which is {1}) is not in C
  + The 2nd element of the bit string is 1, so the 2nd element of U (which is {2}) is in C
* Similarly, the bit string 0000000110 represents {8, 9}
* And the bit string 0101010101 represents {2, 4, 6, 8, 10}

Set operations such as **union**, **intersection** and **complement** may be carried out on bit strings. The simplest way is to use a table and to work **bitwise**, which is to carry out the operations on the first bits, then the second bits, and so on. This is comparable to constructing a truth table that is “turned on its side”.

### Example 2:

* The universal set is U = {4, 6, 9, 13, 18, 25}
* Set A = {6, 13, 18}
* Set B = {4, 13, 18, 25}

Suppose we want to find A ∪ B, A ∩ B and A'. Using a table and performing the operations **bitwise**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **U** | **4** | **6** | **9** | **13** | **18** | **25** |
| **A** | 0 | 1 | 0 | 1 | 1 | 0 |
| **B** | 1 | 0 | 0 | 1 | 1 | 1 |
| **A ∪ B** | 1 | 1 | 0 | 1 | 1 | 1 |
| **A ∩ B** | 0 | 0 | 0 | 1 | 1 | 0 |
| **A'** | 1 | 0 | 1 | 0 | 0 | 1 |

From the table we get:

* A ∪ B bit string is 110111
  + A ∪ B = {4, 6, 13, 18, 25}
* A ∩ B bit string is 000110
  + A ∩ B = {13, 18}
* A' bit string is 101001
  + A' = {4, 9, 25}

### Task:

U = {Argentina, Australia, Belgium, Brazil, Cameroon, Chile, Cuba, Denmark, Egypt, Ethiopia, Fiji, France, Germany, Ghana, Greece, Haiti}

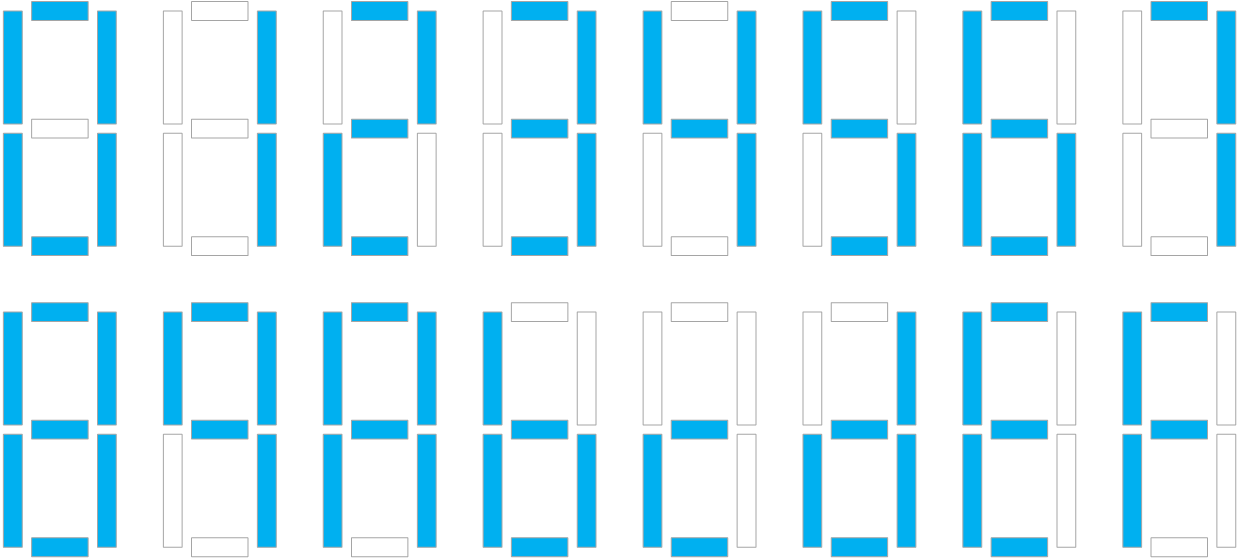
1. Find the bit string that represents the set L = {Brazil, Fiji, Ghana, Greece}
   * Bit string of L is 0001 0000 0010 0110
2. Find the set M represented by bit string 0000 0110 1001 0001
   * M = {Chile, Cuba, Egypt, France, Haiti}
3. Let P, Q, R and S be sets represented by the following bit strings respectively (see table below):
   * Find the bit string that represent the set T = (P ∩ Q') ∪ (R' ∩ S)
     + Bit string for T is 1000 0010 1010 0100
4. List the countries in the set T
   * T = {Argentina, Cuba, Egypt, Fiji, Ghana}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **U** | **Ar** | **Au** | **Be** | **Br** | **Ca** | **Ch** | **Cu** | **De** | **Eg** | **Et** | **Fi** | **Fr** | **Ge** | **Gh** | **Gr** | **Ha** |
| **L** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| **M** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| **P** | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| **Q** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| **R** | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| **S** | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| **Q'** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| **P ∩ Q'** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| **R'** | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| **R' ∩ S** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **T** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

## 

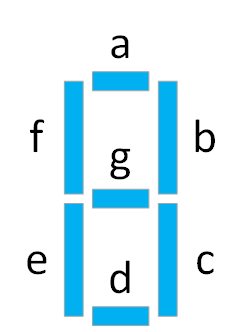
## Question 5

In the lecture notes there is an application of Karnaugh maps to a 7-segment display that is used in some calculators and similar digital devices. The display is sometimes **extended** to include **all** hexadecimal digits. A typical extended display is:



7-seg chars

Notice the display for the hexidecimal digits for 10 through 15. Some are shown as upper case and some lower case: A b c d E F. As in the lecture notes we label each of the seven segments as shown:



7-seg segments

When any of the keys, 0 1 2 3 4 5 6 7 8 9 0 A B C D E F is pressed, a 4-bit binary signal wxyz is generated, as shown in the table below.

For each of the seven segments, there is a corresponding Boolean function and a circuit that has output 0 or 1 as the various keys are pressed. Denote these functions by a(w,x,y,z), b(w,x,y,z), …, g(w,x,y,z).

### Task:

Your task concerns the two segments d and g, and is as follows. For **each** of the two functions d(w,x,y,z) and g(w,x,y,z):

#### 1: Construct a truth table for functions d(w,x,y,z) and g(w,x,y,z)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Key | w | x | y | z | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| A | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| B | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| C | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| D | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| E | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| F | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

#### 

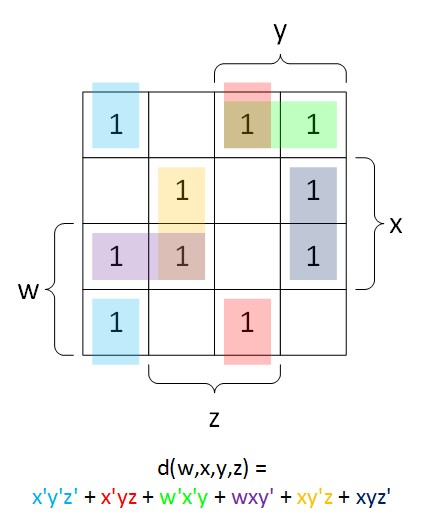
#### 2: Construct the corresponding Karnaugh Map

Use the *simple* labelling on the Karnaugh maps and show your groups clearly.

##### d(w,x,y,z)

Sum of products:

d(w,x,y,z) = w'x'y'z' + w'x'yz' + w'x'yz + w'xy'z + w'xyz' + wx'y'z' + wx'yz + wxy'z' + wxy'z + wxyz'

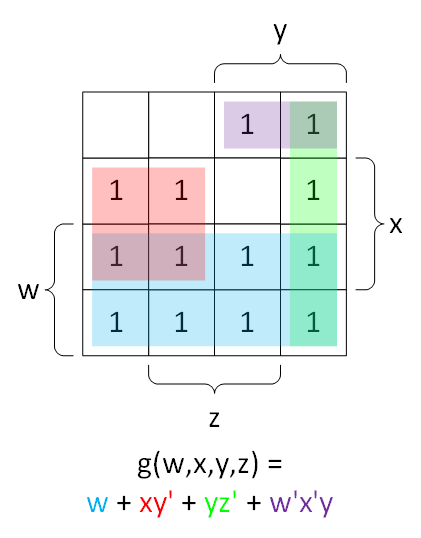


kmap d

##### g(w,x,y,z)

Sum of products:

g(w,x,y,z) = w'x'yz' + w'x'yz + w'xy'z' + w'xy'z + w'xyz' + wx'y'z' + wx'y'z + wx'yz' + wx'yz + wxy'z' + wxy'z + wxyz' + wxyz



kmap g

#### 3. Find a minimal sum of products for the function

d(w,x,y,z) = x'y'z' + x'yz + w'x'y + wxy' + xy'z + xyz'

g(w,x,y,z) = w + xy' + yz' + w'x'y