CSU11021 – Examination

1. Set Symmetric Difference

For the first question, I noted that the set symmetric difference of two sets can be written as $(A \cup B)\ /\ (A \cap B)$, or as $(A \setminus B) \cup (B \setminus A)$. I needed to find the union of both sets and remove the intersecting values.

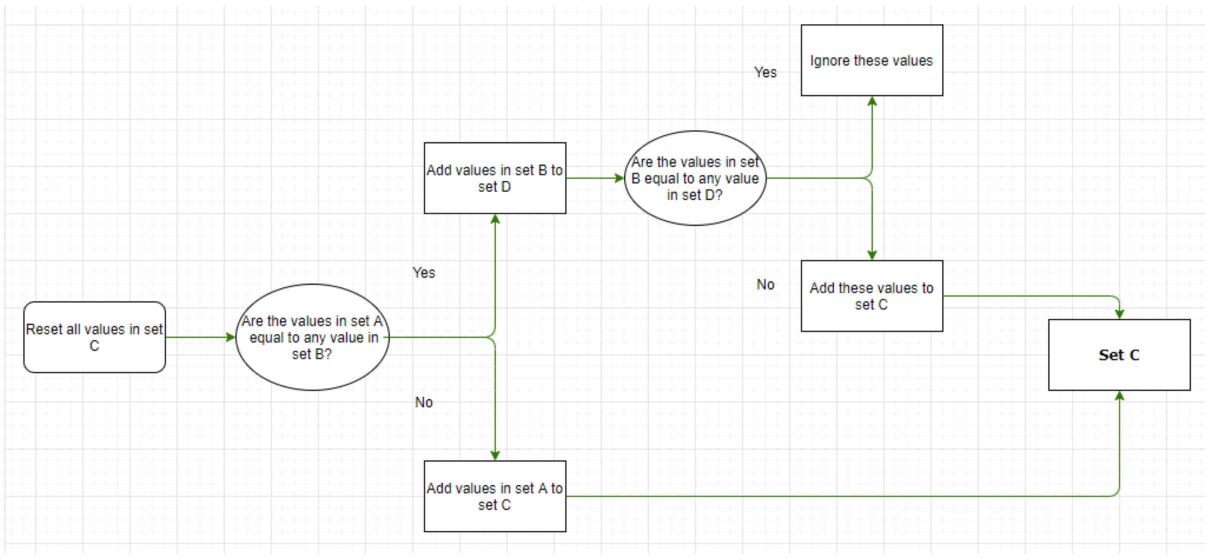
I initially considered using brute force and comparing each value in set A to each value in set B. I would then add the value in set A to set C if it were not also in set B, and then the same for each value in set B. That would be a simple algorithm but would require many nested loops and duplicated code.

I instead decided to add non-duplicated values from set A to set C and add any duplicated values to a set D that I created in memory. For set B if any value is not in set D, I would add it to set C. After every non-duplicated value is added to set C, I replace the first value with the size of set C and put the replaced value at the end. This can be done since the order of values does not matter.

An example: $A = \{4, 21, 15, 7, 39\}, B = \{4, -39, 7, 0, 15\}$

I create a space in memory called “tagged” and assign it to a register. Then I clear each value in set C to make sure it is empty. After assigning two registers to the sizes of both sets A and B (4, 4), I check each value of A to see if it is found anywhere in B. If it is not, I add this value to set C (21, 39). If it is, I “tag” the corresponding value in set B by adding it to newly created space set D (15, 7). I then go through each value in set B and check if it can be found anywhere in set D. If it is not, I add the value in set B to set C (21, 39, -39, 0). If it is, I simply discard it. To get the size, I added one to a register any time I added a value to set C. I add the size to the beginning by putting the first value into a separate register, putting the size in the first position, and putting the first value at the end, creating the final set C (4, 21, 39, -39, 0).

Flowchart:


Reset all values in set C
Are the values in set A equal to any value in set B?
Yes
No
Add values in set B to set D
Add values in set A to set C
Are the values in set B equal to any value in set D?
Yes
No
Ignore these values
Add these values to set C
Set C

Pseudocode:

```
A.size = A[0]
A.size = B[0]
for (int q = 0; q < 64; q++) {
  temp = 0
  C[q] = temp
}

for (p = 0; p < A.size; p++) {
  i = A[p + 1]
  q = 0
  for (q = 0; q < B.size; q++) {
    j = B[q]
    if (i != j) {
    B += 1
    q += 1
  }
  C[p] = i
  D[p] = j
  i ++
  temp = 4
  q *= temp
  B -= B.size
  p += 1
}

q = D.size * temp
D -= q
q = 0
for (q = 0; q < B.size; q++) {
  count = 0
  j = B[0]
  for (count = 0; count < D.size; count++) {
    m = D[0]
    if (m != j)
    count += 1
  }
  C.add(j)
  C.size += 1
  q += 1
  temp = 4
  count = temp * count
  D -= count
}
temp = 4
mainSize = temp * C.size
C -= mainSize
firstVal = C[0]
C[0] = C.size
C += mainSize
C[mainSize - 1] = firstVal
```

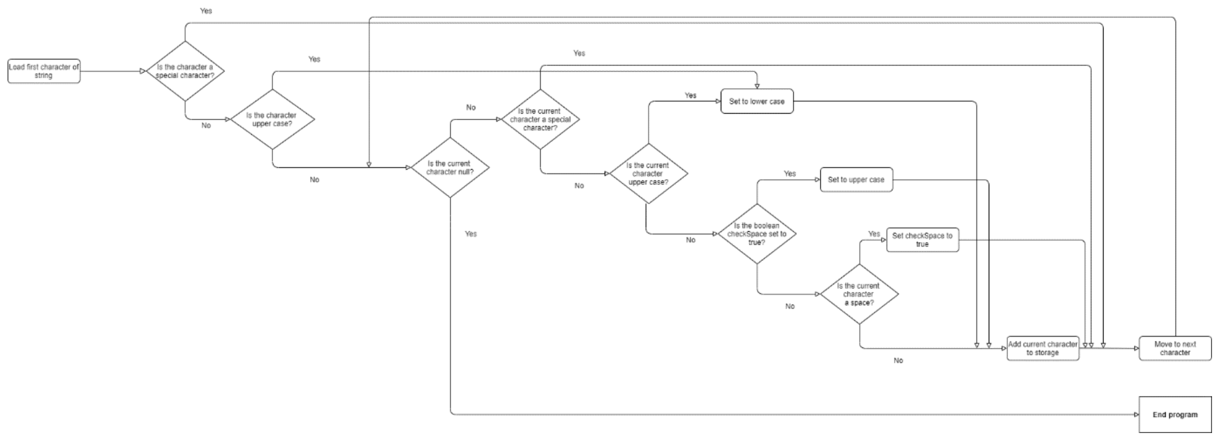
2. lowerCamelCase

For the second question, I noticed that lowerCamelCase is functionally similar to Proper Case, in that you capitalise the first letter of each word. Knowing this, I utilised a similar strategy to deal with this question by also removing any spaces and setting the first letter to lower case.

In terms of code, my approach is a similar but refined version of the Proper Case assignment. I created a Boolean value and assigned it to 0, or ‘false’. This flag represents the state of the previous character; that is, it represents whether the previous character was a space. If the current character is a space, the flag is set, and the current character is not added to R0. If the flag is set, then the current character is set to be upper case, and the flag is unset. If the character is special (period, comma, apostrophe, etc.) then it is also ignored. Given the number of if statements involved, I found that working with a flowchart really helped.

An example: “… Hello world.”
I take the first character of the string (‘.’) and check if it is a special character. If it is, I move onto the next character, but if it is not, I check if it is upper case. If it is, I set it to lower case and move onto the next character.
If it is not, the code begins a loop. If the current character is null, end the program. If it is not, check if it is a special character. If it is, ignore it and move onto the next character. If it is not, check if it is upper case. If it is, set it to lower case and store it. If it is not, check if the boolean checkSpace is set to true. If it is, set the current character to upper case and store it. If it is not, check if the current character is a space. If it is, set checkSpace to true and ignore the current character. If it is not, then it passes all previous tests and can be stored. After this, the program moves onto the next character and starts over.

Flowchart:


Load first character of string
Is the character a special character?
Yes
No
Is the character upper case?
Yes
No
Is the current character null?
No
Yes
Is the current character a special character?
Yes
No
Is the current character upper case?
Yes
Set to lower case
No
Is the boolean checkSpace set to true?
Yes
Set to upper case
No
Is the current character a space?
Yes
Set checkSpace to true
No
Add current character to storage
Move to next character
End program

Pseudocode:

```
i = A[0]
boolean checkSpace = false
if i < 'A'
    pass
if i > 'z'
    pass
if i => 'A' && i <= 'Z' {
    makeLowerCase(i)
    B.add(i)
}
for (p = 0; p < A.size, i++) {
    if (A[p] != ' ') {
        if (A[p] < 'A') {
            continue
        }
        if (A[p] > 'z')
            continue
        }
        if (A[p] > 'A' && A[p] < 'Z') {
            A[p] += 0x20
            B.add(A[i])
        }
        if (checkSpace == true) {
            A[p] -= 0x20
            checkSpace == false
            B.add(A[p])
        }
        if (A[p] == ' ') {
            checkSpace == true
            continue
        }
    }
    B.add(A[p])
}

function makeLowerCase(char i) {
    i += 0x20
}
```