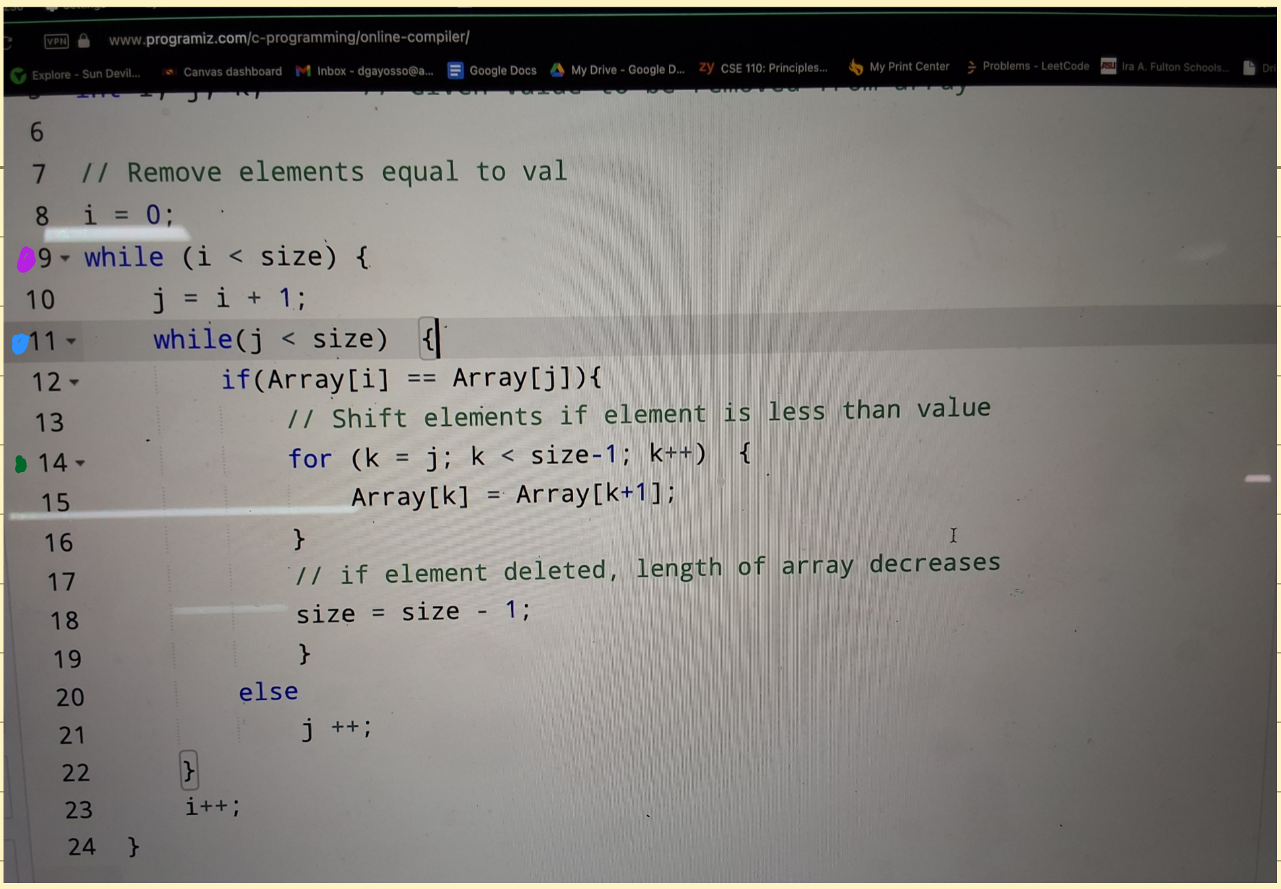
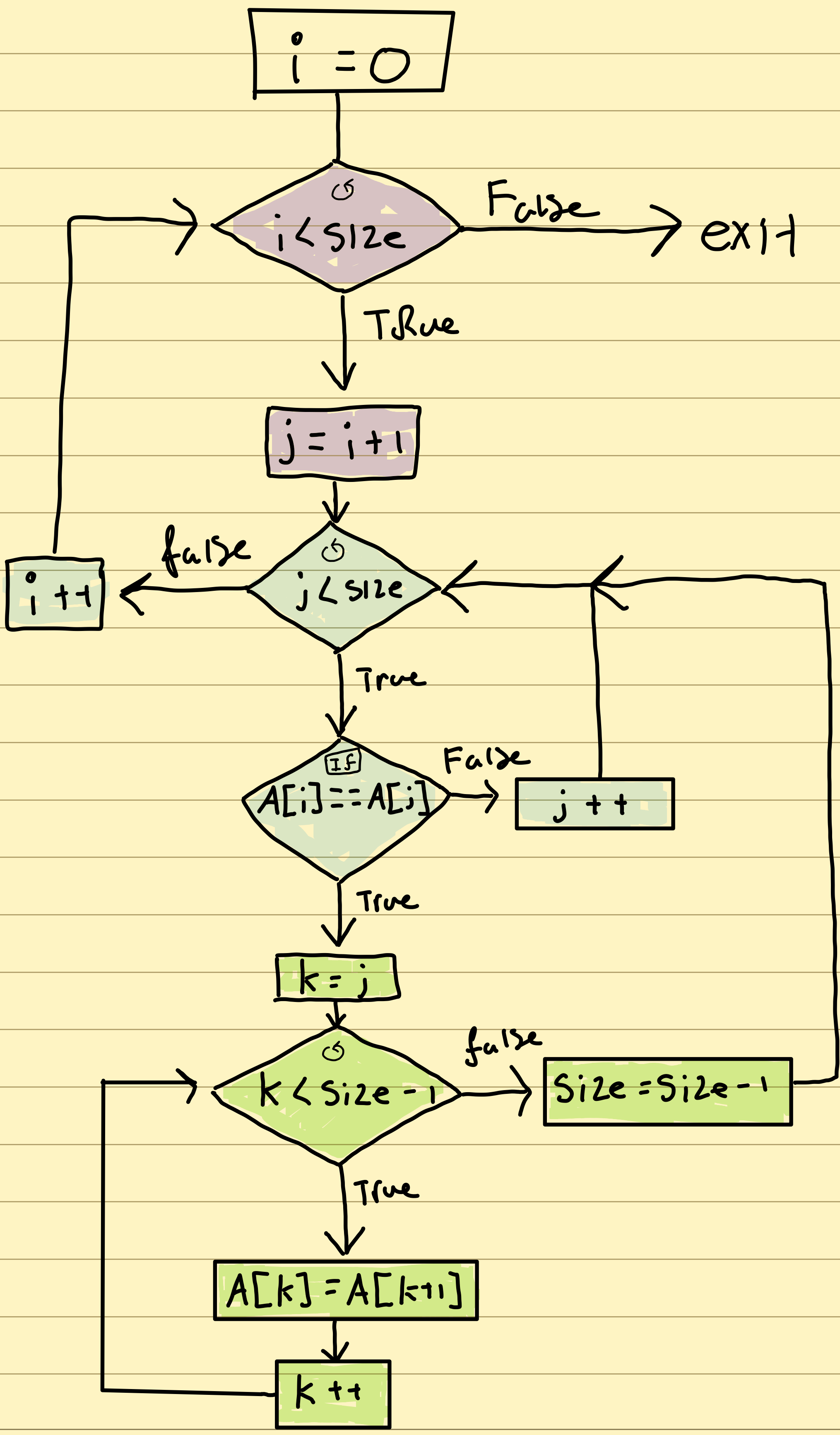


```

// Remove elements equal to val
i = 0;
while (i < size) {
      j = i + 1;
      while(j < size)  {
          if(Array[i] == Array[j]){
              // Shift elements if element is less than value
              for (k = j; k < size-1; k++)  {
                  Array[k] = Array[k+1];
              }
              // if element deleted, length of array decreases
              size = size - 1;
              }
          else
              j ++;
      }
      i++;
 }
```

Outer loop
Inner loop
for loop



Registers	Data	Variable
$ S0	x	A[0]
$ S1	y	size
$ S2	0	i
$ S3	i + 1	j
$ S4	j	k
$ S5		
$ S6		
$ S7		

- a Pointer, to A[0]
- the size of the array

} Int, used in loops

Registers	Data/use
$ t0	1:0
$ t1	i×4 } A[i] ‖ k·4 } A[k]
$ t2	j×4 } A[j] ‖ [k+1]·4 } A[k+1]
$ t3	size - 1
$ t4	
$ t5	
$ t6	
$ t7	
$ t8	
$ t9	

- Holds T:F, values for all branch calls

} used for calculating offset and array location in memory

- used during the for-loop, and it's reuseable, when exiting

size = size-1

$s_1 = $t_3