In computer science, **Compare-And-Swap** (**CAS**) is an atomic instruction used in multithreading to achieve **synchronization**. It compares the contents of a memory location with a given value and, only if they are the same, modifies the contents of that memory location to a new given value.

Usually we use the version or a timestamp to be a marker. Before we handle our service,we take the version or timestamp. When we finish our service, we need to check the marker. If the version or the timestamp keep the same, we store our data to service and increase version by one. Otherwise, execute the instruction from beginning.

```
int compare-and-swap(int *value,int expectedValue,int newValue){

        Int temp = *value;

        If(exceptedValue == *value){

        *value = newValue;
    }

    return temp;
}
```

In this example, we compare the original value: temp = *value;

If (If(exceptedValue == *value) means no other threads change the value,

Assign *value to a newValue. Otherwise, execute the instruction again.

There is another similar one: **Test-and-Set (TAS)**

```
boolean test-and-set(boolean *target){
  boolean rv = *target;
  *target = true;
return target*;
}

Do {
    while(test-and-set(&lock));
      //Critical section
      lock = false;
      //other codes
}
while(true)
```

We don’t use a lock, but we achieve synchronization as well.