Assignment 4:

Exercise 1:

Query 1 :

```
match= SASE({schema=[['auction1','Integer'],['tmpstamp1','STARTTIMESTAMP'],
auction2','Integer'],['tmpstamp2','STARTTIMESTAMP']],
type='Result',
query='PATTERN SEQ(bid auc1, bid auc2)
WHERE skip_till_next_match(auc1,auc2)
{auc1.auction = 0 and auc2.auction = 1}
WITHIN 10 seconds
RETURN auc1.auction, auc1.timestamp, auc2.auction, auc2.timestamp'}, bid)
```

Query 2:

```
match= SASE({schema=[['auction1','Integer'],['tmpstamp1','STARTTIMESTAMP'],
['price1','Double'],['auction2','Integer'],
['tmpstamp2','STARTTIMESTAMP'],['price2','Double']],
type='Result',
query='PATTERN SEQ(bid auc1, bid auc2)
WHERE skip_till_next_match(auc1,auc2)
{[auction] and auc2.price > auc1.price} WITHIN 10 seconds
RETURN auc1.auction, auc1.timestamp, auc1.price, auc2.auction, auc2.timestamp,
auc2.price'}, bid)
```

Query 3:

```
match= SASE({schema=[['auction','Integer'],['tmpstamp','STARTTIMESTAMP'],
['price','Double'],['auction1','Integer'], ['tmpstamp1','STARTTIMESTAMP']
,['price1','Double']],
type='Result',
query='PATTERN SEQ(bid auc,bid+ a[]) WHERE skip_till_next_match (auc,a[])
{[auction] and auc.price = 0.5 * a[a.LEN].price}
WITHIN 60 seconds RETURN auc.auction, auc.timestamp, auc.price, a[a.LEN].auction,
a[a.LEN].timestamp, a[a.LEN].price'}, bid)
```

Query 4:

```
match= SASE({schema=[['auction','Integer'],['tmpstamp','STARTTIMESTAMP'],
['price','Double'],['auction1','Integer'], ['tmpstamp1','STARTTIMESTAMP']
,['price1','Double']],
type='Result',
query='PATTERN SEQ(bid auc,bid+ a[]) WHERE skip_till_next_match (auc,a[])
{[auction] and auc.price + (auc.price * 0.8)<a[a.LEN].price}
WITHIN 10 seconds RETURN auc.auction, auc.timestamp, auc.price, a[a.LEN].auction,
a[a.LEN].timestamp, a[a.LEN].price'}, bid)
```

Query 5:

```
match = SASE({SCHEMA = [['person1', 'integer'],['person2', 'integer'],['person3', 'integer']],
QUERY = 'PATTERN SEQ(person person1,person person2,person person3)
WITHIN 30 seconds
RETURN person1.id, person2.id, person3.id', TYPE = 'result'}, person)
```

Query 6:

```
match = SASE({SCHEMA = [['person_id_1', 'integer'],['auction_id_1', 'integer'],['person_id_2',
'integer'],['auction_id_2', 'integer'],['person_id_3', 'integer'],['auction_id_3', 'integer']],
QUERY = 'PATTERN SEQ(person person1, auction auction1,person person2, auction auction2,
person person3, auction auction3)
where skip_till_any_match(person1,auction1,person2,auction2,person3,auction3)
{person1.id = auction1.seller and person2.id = auction2.seller and person3.id =
auction3.seller}
within 120 seconds
return person1.id, auction1.id, person2.id, auction2.id, person3.id, auction3.id', TYPE =
'result'}, person,auction)
```

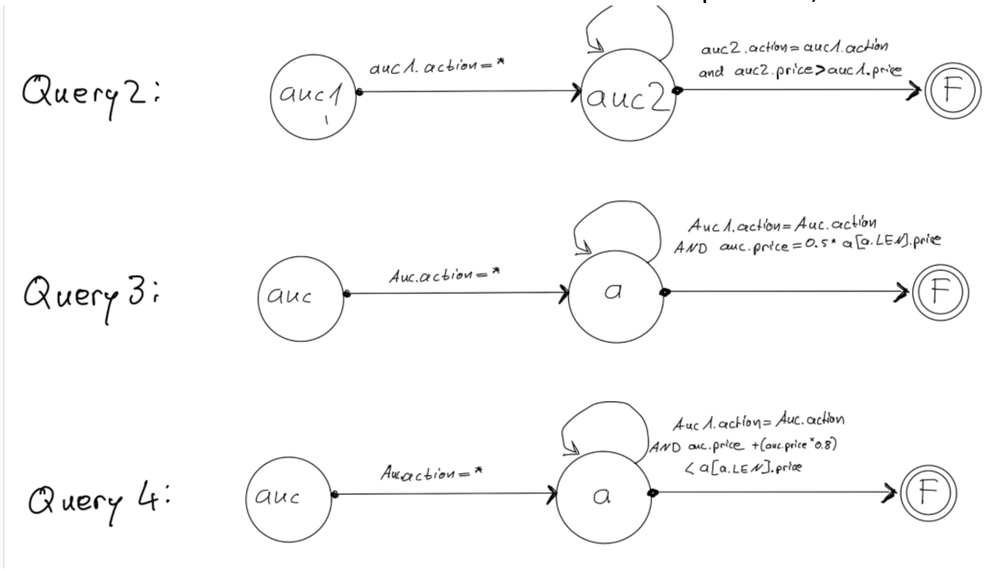
Query 7:

```
PATTERN SEQ(nexmark:person2 person, nexmark:auction2 auction, nexmark:bid2+ bid[])
where skip_till_any_match(person, auction, bid[])
{person.id = auction.seller and auction.id = bid[1].auction and bid[1].auction = bid[i].auction
and count(bid[..i-1].auction)=5 within 10 minutes
return person.name, auction.id, count(bid[..i-1].auction)
```

Query 8:

```
PATTERN SEQ(nexmark:person2 person, nexmark:auction2 auction, nexmark:bid2 bid)
where skip_till_next_match(person, auction, bid)
{person.id = auction.seller and bid.bidder = person.id
return person.name, auction.id)
```

9. Draw the automation and state transitions for the queries 2, 3 and 4.



Exercise 2:

1. What are the pattern policies offered by SASE+?
    a. Strict contiguity
       Means that all Events are immediately contiguous so that there is no event in the stream which does not match a evaluation
    b. Skip till any match
       All events that do not match are skipped until the next relevant event. This is a event which matches a evaluation.
    c. Skip till next match:
       Same as skip till any but now nondeterministic actions are possible on the relevant events.
    d. Partition contiguity:
       Events are immediately contiguous in the same partition
2. Give a use case for each pattern policy (use case means an example where the policy is applied)
    a. Strict contiguity
       Matching Strings in regular Expression e.g. Email verification
    b. Skip till any match:
       Waiting for offers with higher bid than previous
    c. Skip till next match:
       Waiting for offers with higher bid than previous with additionally allowing non deterministic actions on relevant events
    d. Partition contiguity:
       Sorting of many items based on their parent category.
3. Explain briefly what each of the clauses PATTERN, WHERE, and WITHIN is responsible for.

   PATTERN clause: specifies a sequence pattern, which pattern should be matched.
   WHERE clause: imposes value-based constraints on the events addressed by the pattern.
   WITHIN clause: specifies a time window over the entire pattern.

4. What are the termination criteria for a *Kleene* closure (might need to look further than the slides content for this, you can take a look at the paper: *On Supporting Kleene Closure over Event Streams*; Daniel Gyllstrom, Jagrati Agrawal, Yanlei Diao and Neil Immerman).

   Termination criteria for contiguity requirement: Kleene closure terminates if the next event in the input for a given partition does not satisfy the relevant predicates
   Termination criteria for relaxed pattern policies:
   Time constraint with WITHIN clause: The time windows for the entire pattern is also used for every Kleene plus in this pattern.
   Minimal effort: Kleene closure is allowed to perform minimal computation
and
   then break
5. Which additional general pattern policy is „hidden“ in the termination criteria of Kleene closure in SASE+?

Skip until next match: All irrelevant events will be discarded until the next relevant event comes in. With this we try to find all relevant events to the sequence until the termination criteria is fulfilled