

Question 3. (23 = 16 + 6 + 3 points) Suppose that there are two global channels defined:

```
val reqCh : (int * int ivar) chan    (* request channel *)
val updCh : int chan                (* update channel *)
```

For this question, you will define three functions with signatures

```
val addServer : int -> unit    (* RPC server constructor *)
val addn      : int -> int     (* client call #1      *)
val update    : int -> unit    (* client call #2      *)
```

and with the following behavior:

- The call `addServer n` creates an “add n ” server, with local state n , that accepts requests on the two global channels. An *add* request of the form (i, iv) on `reqCh` stores the value of $i+n$ into the I-variable iv and leaves the server state unchanged. An *update* request of the form m on `updCh` turns the server into an “add m ” server (i.e., its local state becomes m).
- The call `addn i` computes and returns $i + n$ by making an add request on `reqCh` to the server (n is the local state of the server)
- The call `update m` makes an update request on `updCh` to the server, turning it into an “add m ” server.

You may assume that the `CML` and `SynchVar` structures have been opened, and you are **not** allowed to use the `MakeRPC` structure.

Question 4. (14 = 7 × 2 points) For each of the following CML primitives, indicate its type by writing it next to its name:

CML.sendEvt

CML.recvEvt

CML.wrap

CML.guard

CML.withNack

CML.choose

CML.timeOutEvt

Question 5. (21 = 7 × (1 + 2) points) For each of the following CML primitives, indicate both its type and whether or not it can block when called (circle Y for “can block” or N for “does not block”):

Y N CML.spawn

Y N CML.send

Y N CML.recv

Y N CML.sendPoll

Y N CML.recvPoll

Y N CML.synch

Y N SynchVar.iPut

Y N SynchVar.iGet

Y N SynchVar.mTake

Y N SynchVar.mSwap

Y N Mailbox.send

Y N Mailbox.recv

Y N Multicast.multicast

Y N Multicast.recv

