qb_preemptcmp($runningjob, $candidatejob, $host)

This is a comparison function which is slightly different from the previous cmp functions. This routine is passed the job which the system plans to replace $runningjob and the $host it is running on. The routine is also passed the $candidatejob which is could replace the $runningjob. If developer wants to cancel the preemption, the function must return the value 0, to use the default policy set in the supervisor_preemption_policy setting the function must return a -1. To aggressively preempt the job, the function must return a 2 and for passive preemption the function must return a 1.

1. 0 = Skip preemption
2. -1 = Use default preemption setting
3. 1 = Passively preempt job
4. 2 = Aggressively preempt job

The format of the input parameters for jobs and hosts are standard Perl hash structures.

```
$job = {
"id" => 1000,

"pid" => 1,
"pgrp" => 1000,
"priority"=> 2,
"user" => "username", "status" => "pending", "name" => "my job’s name", "label" => "qube1", "cluster" =>
"/my/cluster/", "cpus" => 2,
"prototype" => "cmdline", "requirements" => "", "reservations" => "", "restrictions" => "", "account" =>
"shotname",

};
$host = {

"name" => "host’s name ",
"state" => "active",
"cluster" => "/my/cluster", "resources" => "host.processors=10", "restrictions" => "",

"address" => "192.168.10.22"

};
```

Important: If you implement this routine in a custom algorithm and mix passive and aggressive preemptions, make sure that the qb_supervisor_preempt_policy is set to "mixed".
See supervisor_preempt_policy for details.