Calling sequence for main filter assimilation program using rough DART class specifications

```
! Initialize the model and the obs_sequence
obs_sequence = init_input_obs_sequence(file_name)
model_size = initialize_assim_model
! Allocate space for everything that depends on model size

! Get a model initial condition
state = get_initial_condition()

! Generate an ensemble (where does the info for pert cov come from?)
ens(:) = state + pert(:)

! Initialize output for state variable space output
file_id = init_diag_output(‘output_state_file’, ‘Square root filter diagnostic output’, ens_size * 2, ens_meta_data_text)

! Initialize output for observation space output
obs_sequence_out = init_output_obs_sequence(‘output_obs_file’,  ‘Observation space output’, 2*ens_size + 1, ‘Square root filter diagnostic observation output’, obs_meta_data_text)

! Loop through all the observation sets
while not(end_of_sequence(obs_sequence))

! Get next obs_set
obs_set = get_next_obs_set(obs_seqeuence)

! Compute time at which we wish to do assimilation
assim_time = get_closest_state_time_to(get_obs_set_time(obs_set))

! Advance all ensemble members to this time
ens(:) = advance_state(ens(:), assim_time)

! Output the prior ensembles
something = output_diagnostics(file_id, ens(:), assim_time, :)

! Generate ensemble prior for obs in this set
num_obs = get_total_num_obs(obs_set)

! Allocate storage for prior ensembles for this number of observations
allocate(ens_obs(num_obs, ens_size))

! Compute ensemble priors
ens_obs(:) = get_expected_obs(obs_set, ens(:))

! Output the prior ensemble obs fields
! Write first field and header
something = output_obs_set(output_obs_seqeuence, obs_set)
! Next write all the prior ensemble copies of this obs_set
loop through ensembles
   temp_obs_set = set_obs_set_values(obs_set, ens_obs(:)
   something = output_obs_set(output_obs_sequence, temp_obs_set, index)
end loop through ensembles

! Get the observational covariance (diagonal for now)
obs_err_cov = get_diag_obs_err_cov(obs_set)

! Get the observations
obs = get_obs_value(obs_set)

! Get the close states lists
num_close_states = get_num_close_states(obs_set, cutoff_radius)
! Allocate some storage if needed or do whatever
close_states_list = get_close_states(obs_set, cutoff_radius)

! Loop through each observation in the set, compute deltas
do i = 1, num_obs
   delta(:) = compuation(obs_err_cov(:), ens_obs(:), obs(:))

! Loop through each close state variable for an observation and update
   do j = 1, num_close_states
      do regression between this state variable and this ob and update
   end do j
end do i

! Output the posteriors for both the obs_set and the state ensembles as above

end outer loop
```

How does DART generate observation data files for synthetic observations; Uses an observations definition file that contains no observations (only defs)

1. Get time of next observation set (avoid time interpolation for now)
2. Advance state in time to the time of this observation set using model
3. Output the true state at this time
3. Get true observation values using Observation_set
4. Output the true observations

4. Get the observational error covariance (assume diagonal for now to make life easy)
5. Add a random sample from the error covariance to the obs
6. Output these synthetic observations
7. Cycle through this