```
###!/usr/local/bin/env python

# I made the following changes to this test case:
#
# 1) Changed the test system from harmonic oscillators to the alanine dipeptide
# 2) Changed the number of MC iterations
# 3) Changed select (internal) function names

# ============================================================================================
# GLOBAL IMPORTS
# ============================================================================================

# Non-scientific python packages needed for this protocol

import os
import sys
import timeit
from io import StringIO
import openmmtools as mmtools
from openmmtools import testsystems
from simtk import unit

# This is where replica exchange utilities are imported from Yank

from yank import mpi
from yank.multistate import MultiStateReporter, MultiStateSampler, ReplicaExchangeSampler,
ParallelTemperingSampler, SAMSSampler
from yank.multistate import ReplicaExchangeAnalyzer, SAMSAnalyzer
from yank.multistate.multistatereporter import _DictYamlLoader
from yank.utils import config_root_logger
from yank.commands import analyze

# quiet down some citation spam
MultiStateSampler._global_citation_silence = True

# ============================================================================
# RUN REPLICA EXCHANGE
# ============================================================================

def run_replica_exchange(verbose=False, verbose_simulation=False):

    sampler_states = list()
    thermodynamic_states = list()

    # Define thermodynamic states.
    temperatures = [250.0, 300.0, 350.0, 400.0, 450.0] * unit.kelvin  # Temperatures.
    for temperature in temperatures:
        testsystem = testsystems.AlanineDipeptideVacuum()
        # Create thermodynamic state and save positions.
        system, positions = [testsystem.system, testsystem.positions]
        sampler_states.append(mmtools.states.SamplerState(positions))
        thermodynamic_states.append(mmtools.states.ThermodynamicState(system=system,
        temperature=temperature))

    # Create and configure simulation object.
    move = mmtools.mcmc.LangevinDynamicsMove(timestep=2.0*unit.femtoseconds,
                                             collision_rate=20.0/unit.picosecond,
                                             n_steps=100, reassign_velocities=True)
    simulation = ReplicaExchangeSampler(mcmc_moves=move, number_of_iterations=20)

    # Define file for temporary storage.
    if os.getcwd() == os.getcwd():
        storage = os.path.join(os.getcwd(), 'test_storage.nc')
        if os.path.exists(storage): os.remove(storage)
```

```
        reporter = MultiStateReporter(storage, checkpoint_interval=1)
        simulation.create(thermodynamic_states, sampler_states, reporter)

        config_root_logger(verbose_simulation)
        simulation.run()

        # Clean up.
        del simulation

    if verbose:
        print("PASSED.")

# ==============================================================================
# MAIN
# ==============================================================================

if __name__ == "__main__":
    # Configure logger.
    config_root_logger(False)

    start_time = timeit.default_timer()
    run_replica_exchange()
    stop_time = timeit.default_timer()
    print("Calculation time was: "+str(stop_time-start_time)+" seconds.")

    analyze.dispatch_extract_trajectory({'--checkpoint':
    'test_storage_checkpoint.nc','--fulltraj': True,'--netcdf': 'test_storage.nc',
    '--output': 'traj1','--replica': 1,'--trajectory': 1,'--state': 1,'extract-trajectory':
    True,'--discardequil': False,'--distcutoff': None,
    '--end': None,'--energycutoff': None,'--format': None,'--serial': None,'--skip':
    None,'--skipunbiasing': True,'--start': None,
    '--imagemol': None,'--nosolvent': None,'--verbose': False})
```