REPORT OF QUESTION 1

The code implements mergesort using threads, processes and normal sequential methods and for an array of small length, I have compared it to selection sort also.

FUNCTIONS

- selection_sort()
    - Implements selection sort by taking the array as an argument and the end points of the array and sorts it using the selection sort. The basic logic of selection sort it to swap the smallest element while traversing through the array
- combine()
    - The merge functions for mergesort which merges the two sorted halves of the array
- normal_mergeosrt()
    - Calling a function for sorting the array using mergesort and it makes recursive calls to sort the left half and the right half of the array and merge the sorted halves
- concurrent _mergesort()
    - First, a process if forked and the left half of the array is passed for the process to sort
    - Then in the parent else condition another process is forked and the right half of the array is passes for it to sort
    - Then the parent waits for both the processes to finish and then merges the array by calling combine()
- thread_func()
    - The function which is called by the threads where the argument is passed in structure thread_arg and the parent creates two threads and passing left and right half of the array respectively to both the threads and waiting for these threads to finish and then merging the array
- main()
    - Main just call the respective array functions and calculates their time taken using time.h library functions

FINDINGS

- For n = 5
  - Time taken by concurrent mergesort: 0.000568210
  - Time taken by threaded mergesort: 0.000422003
  - Time taken by normal mergesort: 0.000002113
- For n = 10
  - Time taken by concurrent mergesort: 0.000408781
  - Time taken by threaded mergesort: 0.000254983
  - Time taken by normal mergesort: 0.000000916
- For n = 100
  - Time taken by concurrent mergesort: 0.006080436
  - Time taken by threaded mergesort: 0.003527961
  - Time taken by normal mergesort: 0.000050508
- For n = 2000
  - Time taken by concurrent mergesort: 0.056964196
  - Time taken by threaded mergesort: 0.020828157
  - Time taken by normal mergesort: 0.000868495
- For n = 10000
  - Time taken by concurrent mergesort: 0.512876284
  - Time taken by threaded mergesort: 0.203122849
  - Time taken by normal mergesort: 0.008917799
- For n = 50000
  - Time taken by concurrent mergesort: 3.021100406
  - Time taken by threaded mergesort: 0.823172469
  - Time taken by normal mergesort: 0.059678094

As we can see for small values concurrent and threaded mergesort has a similar time but as size increases, concurrent mergesort becomes slower
And hence the time taken in order is

Concurrent > Threaded > Sequential