1. Issues encountered sorting large arrays and vectors with recursion. How did you fix the issue(s) encountered?

Answer:- Recursion sort causes memory location on a stack .Thus it repeatedly keeps on calling the function till the task is complete. Thus issue encountered was stack overflow exception. This occurred when for large sized array and vector the number of elements exceeded 5000 which resulted in memory crash.

Fixing the issue :- The issue was resolved by increasing the stack reserve size to hold a larger capacity of data.

**Properties -> Configuration Properties -> Linker -> System -> Stack Reserve Size**

Enter a new value, say, 3000000 in the box and click on apply.

-Provided by Anvit

2. The difference in performance between sorting with recursion version iteration. Which performed better, and **why**?

Answer. The performance with recursion is better as the visual studio compiler optimizes recursion. The actual reason for the slower performance is heavy push-pop of the registers in the ill level of each recursive call. Recursion uses system stack to accomplish it 's task. As stack uses LIFO approach and when a function is called the controlled is moved to where function is defined which has it is stored in memory with some address, this address is stored in stack

3. The difference in performance between sorting C-Style arrays versus the STL vector. Which performed better, and **why**?

Answer. C-Style arrays perform better. A STL vector has a number of attributes which potentially makes it slower than a C array:

* A vector does not store its data in a continuous chunk of memory. Pointers to the start and end of the vector will be stored in a different location from the data itself.
* An STL vector always allocates memory for its data on the heap. Heap allocation is slower than stack allocation.

Thus array is like a stack a fixed memory location where as vector is more of like a function call. Thus every time vector is used there is time consumption is form of function call which increases the overall execution time.