Name: \_\_\_\_Tailang CAO\_\_\_\_

The main file is executing file01 and file02, which are each 4kb big, and too small. The runtime is thus too short to be recorded with two decimal numbers.						
Flat pr	ofile:					
Fach co	mnla aquat		1nds			
	-		r seconus.	001f	+0+0]	
			00110			name
				-	-	
		0.54	2			sort1(std::cxx
15.44	2.19	0.53	38948	0.00	0.00	search1(std::c
13.99	2.67	0.48	379465206	0.00	0.00	bool std::opera
11.36	3.06	0.39	647482750	0.00	0.00	bool std::opera
10.64	3.42	0.37	108224639	0.00	0.00	std::char_trait
0.73	3.45	0.03	1	0.03	0.03	_GLOBALsub_I
0.00	3.45	0.00	38931	0.00	0.00	void std::swap <c< th=""></c<>
0.00	3.45	0.00	2	0.00	1.20	find_print_add_r
0.00	3.45	0.00	2	0.00	0.00	
0.00	3.45	0.00	1	0.00	0.00	_GLOBALsub_I
0.00	3.45	0.00	1	0.00		static_initial
0.00	3.45	0.00	1	0.00		static_initial
	Flat pr  Each sal % c time 32.35 15.74 15.44 13.99 11.36 10.64 0.73 0.00 0.00 0.00 0.00 0.00	Flat profile:  Each sample count % cumulative time seconds 32.35 1.11 15.74 1.65 15.44 2.19 13.99 2.67 11.36 3.06 10.64 3.42 0.73 3.45 0.00 3.45 0.00 3.45 0.00 3.45 0.00 3.45	Flat profile:  Each sample counts as 0.02 % cumulative self time seconds seconds 32.35 1.11 1.11 15.74 1.65 0.54 15.44 2.19 0.53 13.99 2.67 0.48 11.36 3.06 0.39 10.64 3.42 0.37 0.73 3.45 0.03 0.00 3.45 0.00 0.00 3.45 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 3.45 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0	Flat profile:  Each sample counts as 0.01 seconds.  % cumulative self  time seconds seconds calls 32.35 1.11 1.11 647482750 15.74 1.65 0.54 2 15.44 2.19 0.53 38948 13.99 2.67 0.48 379465206 11.36 3.06 0.39 647482750 10.64 3.42 0.37 108224639 0.73 3.45 0.03 1 0.00 3.45 0.00 38931 0.00 3.45 0.00 2 0.00 3.45 0.00 1	runtime is thus too short to be recorded with two flat profile:           Each sample counts as 0.01 seconds.           % cumulative self self time seconds seconds calls s/call 32.35 1.11 1.11 647482750 0.00 15.74 1.65 0.54 2 0.27 15.44 2.19 0.53 38948 0.00 13.99 2.67 0.48 379465206 0.00 11.36 3.06 0.39 647482750 0.00 11.36 3.06 0.39 647482750 0.00 10.64 3.42 0.37 108224639 0.00 0.73 3.45 0.03 1 0.03 0.00 0.73 3.45 0.03 1 0.03 0.00 0.73 3.45 0.00 38931 0.00 0.00 0.00 3.45 0.00 2 0.00 0.00 0.00 3.45 0.00 2 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 0.00 0.00 0	Flat profile:  Each sample counts as 0.01 seconds.  % cumulative self self total time seconds seconds calls s/call s/call 32.35 1.11 1.11 647482750 0.00 0.00 15.74 1.65 0.54 2 0.27 0.51 15.44 2.19 0.53 38948 0.00 0.00 13.99 2.67 0.48 379465206 0.00 0.00 11.36 3.06 0.39 647482750 0.00 0.00 10.64 3.42 0.37 108224639 0.00 0.00 0.00 0.73 3.45 0.03 1 0.03 0.03 0.03 0.00 3.45 0.00 38931 0.00 0.00 0.00 0.00 0.00 3.45 0.00 2 0.00 0.00 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 0.00 0.00 3.45 0.00 1 0.00 0.00 0.00 0.00 0.00 0.00 0

	^L						
	Call graph (explanation follows)						
	granula	rity: 0	each sam	ple hit o	covers 2 byte(s	s) for 0.29% of 3.45 seconds	
	index %	6 time	self	children	n called	name <spontaneous></spontaneous>	
	[1]	99.3	0.00	3.42		main [1]	
			0.00	2.40	2/2	find_print_add_records(std	
			0.54 0.00	0.48 0.00	2/2 2/2	<pre>sort1(std::cxx11::basic_ readFile(std::cxx11::basic_</pre>	
					200/0/200/0		
	[2]	69.6	0.53 0.53	1.87 1.87	38948/38948 38948	<pre>find_print_add_records(std search1(std::cxx11::basic_st</pre>	
			0.39		47482750/64748		
			0.00	2.40	2/2	 main [1]	
	[3]	69.6	0.00	2.40		find_print_add_records(std::	
			0.53 	1.87	38948/38948 	search1(std::_cxx11::basi	
			0.39		47482750/64748		
	[4]	54.2	0.39 1.11		547482750 547482750/64748		
	[5]	42.9	1.11 1.11		547482750/64748 547482750	32750 bool std::operator!= <cl< th=""><th></th></cl<>	
	[0]	74.7	0.37		108224639/10822		
			0.54	0.48	2/2	 main [1]	
	[6]	29.7	0.54	0.48	2	sort1(std::cxx11::basic_stri	
			0.48 0.00		379465206/37946 38931/38931	bool std::operator< <cl> void std::swap<char, std::<="" th=""><th></th></char,></cl>	
	[7]	14.0	0.48 0.48		379465206/37946 379465206		
	[8]	10.6	0.37 0.37		108224639/10822 108224639		
			0.03	0.00	1/1	libc_csu_init [10]	
	[9]	0.7	0.03 0.00	0.00 0.00	1/1 1 1/1	_GLOBALsub_IZ7search1PNSt7 static_initialization_an	
	[10]	0.7	0.00	0.03		<spontaneous> libc_csu_init [10]</spontaneous>	
	[10]	0.7	0.03	0.00	1/1	GLOBALsub_IZ7search1P	
			0.00	0.00	1/1	_GLOBALsub_IZ8readFile	
			0.00	0.00	38931/38931	sort1(std::cxx11::basic_	
	[17]	0.0	0.00	0.00	38931	void std::swap <char, std::char<="" th=""><th></th></char,>	
			0.00	0.00	2/2	main [1]	
	[18]	0.0	0.00	0.00	2	readFile(std::cxx11::basic_s	
			0.00	0.00	1/1	libc_csu_init [10]	
	[19]	0.0	0.00	0.00	1 1/1	_GLOBALsub_IZ8readFileNSt7 static_initialization_an	
				0.00 		static_initialization_an	
	[20]	0.0	0.00 0.00	0.00 0.00	1/1 1	_GLOBALsub_IZ8readFile static_initialization_and_de	
					1 		
	[21]	0.0	0.00	0.00 0.00	1/1 1	_GLOBALsub_IZ7search1Pl static_initialization_and_de	
						acactoiniteratizacton_and_de	
Q3	gnii	CAA	ena	hle if<	td:: is cha	ar <char>::value, bool&gt;::</char>	tyne
٦,5	. ——-			_		basic_string <char,\$< th=""><th>,,,,</th></char,\$<>	,,,,
	-			-		vasic_striig <triar,\$< th=""><th></th></triar,\$<>	
				6 of tim			
Q4	The sa	me or	ne as t	he one	consuming t	the most time.	

	gnu_cxx::enable_if <std::is_char<char>::value, bool&gt;::type std::operator==<char>(std::cxx11::basic_string<char,\$< th=""></char,\$<></char></std::is_char<char>					
Q5	Search1 uses 0.53 seconds per call, total call time is 0 second.  This might be because search1 itself consumes a significant runtime, but the functions search1 calls do not contribute significantly to the overall execution time.					
Q6	Self seconds per call: 0.00 seconds  Total seconds per call: 0.60 seconds  This implies that the majority of the time spent in find_print_add_records is due to the functions it calls rather than its own code.					
Q7	The total time was 2.40 seconds contributing from the find_print_add_records child function.					
Q10						

## Q8:

Optimization Level	Total execution time
O0 (default)	3.28 sec
01	2.87 sec
02	1.93 sec
03	1.62 sec

## Q9:

Optimization level	Search1	Sort1
01	69.7%	27.4%
O2	57.2%	45.32%
О3	49.1%	49.3%

## Q11:

Sort function	Search function	Total execution time
Sort1	Search1	3.42 sec
Sort2	Search1	4.58 sec
Sort3	Search1	2.47 sec
Sort1	Search2	0.83 sec
Sort2	Search2	2.76 sec
Sort3	Search2	0.01 sec

Q12:

Sort3 Search2 has the best performance fin terms of execution time

Q13:

(3.42 - 0.01)/3.42 = 99% program enhancement percentage.