Zubeyir Ozdemir

3. Timing: Part 1 (20 Points):

Compile and run the program without any extra optimizations, but with *profiling* for timing:

a.*How for 65536 strings of length 8 how many* ***cumulative seconds*** *did insertionSort() take?*

22.25 seconds

b.*How for 65536 strings of length 8 how many* ***cumulative seconds*** *did quickSort_() take?*

0.01 seconds

4.Timing: Part 2 (20 Points):

Compile and run the program *with* optimization, but with *profiling* for timing:

a.*How for 65536 strings of length 8 how many* ***cumulative seconds*** *did insertionSort() take?*

12.46 seconds

b.*How for 65536 strings of length 8 how many* ***cumulative seconds*** *did quickSort_() take?*

0.01 seconds

5.Algorithm choice vs. Compiler optimization (Points 10):

Which is faster?

- A bad algorithm and data-structure optimized with -O2
- [✓ A good algorithm and data-structure optimized with -O0]

6.Parts of an executable (Points 20):

a) objdump -s -j .rodata assign1-0

```
4010c8 2825642d 2564293a 20002564 006e756d  (%d-%d): .%d.num
4010d8 62657220 6f662073 7472696e 6773006c  ber of strings.l
```

b) objdump -d -j .text assign1-0

```
0000000000400b3e <releaseMem>:
400b3e:     55                  push   %rbp
400b3f:     48 89 e5            mov    %rsp,%rbp
400b42:     48 83 ec 20         sub    $0x20,%rsp
400b46:     e8 35 fc ff ff      callq  400780 <mcount@plt>
400b4b:     48 89 7d e8         mov    %rdi,-0x18(%rbp)
400b4f:     89 75 e4            mov    %esi,-0x1c(%rbp)
400b52:     c7 45 fc 00 00 00 00  movl   $0x0,-0x4(%rbp)
400b59:     eb 23               jmp    400b7e <releaseMem+0x40>
```

```
400b5b:     8b 45 fc              mov    -0x4(%rbp),%eax
400b5e:     48 98                 cltq
400b60:     48 8d 14 c5 00 00 00  lea    0x0(,%rax,8),%rdx
400b67:     00
400b68:     48 8b 45 e8           mov    -0x18(%rbp),%rax
400b6c:     48 01 d0              add    %rdx,%rax
400b6f:     48 8b 00              mov    (%rax),%rax
400b72:     48 89 c7              mov    %rax,%rdi
400b75:     e8 66 fb ff ff        callq  4006e0 <free@plt>
400b7a:     83 45 fc 01           addl   $0x1,-0x4(%rbp)
400b7e:     8b 45 fc              mov    -0x4(%rbp),%eax
400b81:     3b 45 e4              cmp    -0x1c(%rbp),%eax
400b84:     7c d5                 jl     400b5b <releaseMem+0x1d>
400b86:     48 8b 45 e8           mov    -0x18(%rbp),%rax
400b8a:     48 89 c7              mov    %rax,%rdi
400b8d:     e8 4e fb ff ff        callq  4006e0 <free@plt>
400b92:     c9            leaveq
400b93:     c3            retq
```

c) objdump -t -j .bss assign1-0

```
00000000006020a8 g     O .bss     0000000000000004          strLen
```

d) It cannot be found.

7.Compiler optimizations (Points 10):

1) Non-optimized insertionSort has two NULL instructions which do nothing. These useless instructions are eliminated in assign1-2.

```
400c91:     00
400ca8:     00
```

2) The use of push/pop in assign1-2 within main() is more efficient