Pre-Lecture 8

Due Sep 16 at 9am **Points** 10 **Questions** 7 **Available** until Sep 16 at 9am
Time Limit None **Allowed Attempts** 2

Instructions

Take this quiz ***after you have watched the required videos and/or read the associated sections of the textbook***. See **Lecture 8: Optimization I**.

You may attempt this quiz twice. Incorrect responses are marked after each attempt. Correct answers are revealed at the start of class for this lecture.

Carefully note the deadline for responses. Submissions are not accepted after the deadline, and there is no grace period.

This quiz was locked Sep 16 at 9am.

Attempt History

	Attempt	Time	Score
KEPT	**Attempt 2**	10 minutes	10 out of 10
LATEST	**Attempt 2**	10 minutes	10 out of 10
	Attempt 1	1 minute	9.25 out of 10

Score for this attempt: **10** out of 10
Submitted Sep 16 at 2:18am
This attempt took 10 minutes.

Question 1 — 1 / 1 pts

In general, programmers should not expect an optimizing compiler to improve the asymptotic efficiency of a program. However, in cases of known functions, the compiler may make such an improvement.

Correct!

- (•) True
- ( ) False

Question 2

1 / 1 pts

Which of the following is a limitation of optimizing compilers? (Select all that apply.)

Correct!

- [x] As long as program behavior is defined, it must not be changed by the compiler.

Correct!

- [x] Analysis is not based on program input.

Correct!

- [x] Most analysis does not cross procedures or files.

Correct!

- [x] Behavior obvious to the programmer can be obfuscated by coding style.

Question 3

2 / 2 pts

Consider the following procedure written in C:

```
void scale_by_7(int arr[]) {
  int i;
  for(i = 0; i < 100; i++)
    arr[i] *= 7;
}
```

This code has been compiled with -O2 to get the following x86 code:

```
        xorl    %eax, %eax
.L3:
        movl    (%rdi,%rax), %ecx
        leal    0(,%rcx,8), %edx
        subl    %ecx, %edx
        movl    %edx, (%rdi,%rax)
        addq    $4, %rax
        cmpq    $400, %rax
        jne     .L3
        ret
```

What optimization has the compiler performed with the third and fourth assembly instructions?

- code motion
- strength reduction
- common subexpression elimination

Correct!

Question 4

2 / 2 pts

Consider the following procedure written in C:

```
void initialize(int arr[], int x, int y) {
  arr[0] = x*y;
  arr[1] = x*y + 1;
  arr[2] = x*y + x;
}
```

This code has been compiled with -O2 to get the following x86 code:

```
        imull   %esi, %edx
        leal    1(%rdx), %eax
        movl    %edx, (%rdi)
        addl    %esi, %edx
        movl    %edx, 8(%rdi)
        movl    %eax, 4(%rdi)
        ret
```

What optimization has the compiler performed?

- code motion
- strength reduction
- common subexpression elimination

Correct!

Question 5

2 / 2 pts

Consider the following procedure written in C:

```
void initialize(int arr[], int x, int y) {
  int i;
  for(i = 0; i < 100; i++)
    arr[i] = x*y;
}
```

This code has been compiled with -O2 to get the following x86 code:

```
        imull   %edx, %esi
        xorl    %eax, %eax
.L3:
        movl    %esi, (%rdi,%rax)
        addq    $4, %rax
        cmpq    $400, %rax
        jne     .L3
        ret
```

What optimization has the compiler performed?

Correct!

- (•) code motion
- ( ) strength reduction
- ( ) common subexpression elimination

Question 6

1 / 1 pts

For the following C function, an optimizing compiler will be blocked from reducing the two memory lookups at address a to just one lookup because of memory aliasing.

```
float f(int* a, float* b) {
  *b = *a + 3;
  return *b / *a;
}
```

- ( ) True

Correct!

- False

Question 7

1 / 1 pts

For the following C function, an optimizing compiler will be blocked from reducing the two memory lookups at address a to just one lookup because of memory aliasing.

```
float f(int* a, char* b) {
  *b = *a + 3;
  return *b / *a;
}
```

Correct!

- True
- False

Quiz Score: **10** out of 10