Topics: Searching, intro to efficiency concepts

Turn in: Print out the assignment and work it out on paper/ Either scan or photograph the assignment once you're done and upload it to the Dropbox.

Name: Linden Stirk

Section 1: Stepping through code

For the following questions, a search algorithm will be given, as well as the inputs. You will need to act as the human computer and step through the algorithm, one command at a time, recording the changes to the variables and stepping through the flow of the function.

For example:

Function:

```
int FindItem( int arr[], int arraySize, int searchItem )
{
    for ( int i = 0; i < arraySize; i++ )
    {
        if ( arr[i] == searchItem )
        {
            return i;
        }
    }

    return -1;
}
```

Inputs:

```
int pos = FindItem( { 1, 3, 5, 7 }, 4, 5 );
```

0 1 2 3

So, the function is being called, with the array:

Index	0	1	2	3
Element	1	3	5	7

And the array size is 4, and the item we're searching for is 5. So then we step through each line…

Step-thru:
Function begin arr[] = { 1, 3, 5, 7 } arraySize = 4 searchItem = 5

For loop begin i = 0
arr[i] == searchItem? arr[0] = 1, searchItem = 5 FALSE

Loop continues i = 1
arr[i] == searchItem? arr[1] = 3, searchItem = 5 FALSE

Loop continues i = 2
arr[i] == searchItem? arr[2] = 5, searchItem = 5 TRUE
Value of *i* is returned

FindItem returns 2.

1. For the given algorithms, record all variable values & changes as you step through the code, one line at a time. If there is a **cout** or **return**, you should also specify what is outputted or returned.

a.

```
for ( int i = 0; i < 3; i++ )
{
    cout << "hi " << i;
}
```

(___/2)

For loop begins i = 0

Message displayed: hi 0

i = 0+1

For loop continues i = 1

Message displayed: hi 1

i = 1+1

For loop continues i = 2

Message displayed: hi 2

i = 2+1

b. (___/2)

```
for ( int i = 0; i < 5; i++ )
{
    if ( i % 2 == 0 )
    {
        cout << i << " even" << endl;
    }
    else
    {
        cout << i << " odd " << endl;
    }
}
```

For loop begins **i =** 0
Is i % 2 == 0? **True / False**
Message displayed: Even

For loop continues **i =** 1
Is 1%2 ==0? False

For loop continues **i =** 2
Is 2%2==0? True

For loop continues **i =** 3
Is 3%2 ==0? False

For loop continues **i =** 4
Is 4%2==0? True

c. (___/2)

```
for ( int i = 0; i < 3; i++ )
{
    for ( int j = 0; j < 3; j++ )
    {
        cout << i * j << endl;
    }
}
```

i · j

Outer for loop begins **i** = 0
Inner for loop begins **i** = 0 **j** = 0 0 · 0 = 0
Message displayed: 0

Inner loop continues **i** = 0 **j** = 1 0 · 1 = 0
Message displayed: 0

Inner loop continues **i** = 0 **j** = 2 0 · 2 = 0
Message displayed: 0

Outer loop continues **i** = 1
Inner for loop begins **i** = 1 **j** = 0 1 · 0 = 0
Message displayed: 0

Inner loop continues **i** = 1 **j** = 1 1 · 1 = 1
Message displayed: 1

Inner loop continues **i** = 1 **j** = 2 1 · 2 = 2
Message displayed: 2

Outer loop continues **i** = 2
Inner for loop begins **i** = 2 **j** = 0 2 · 0 = 0
Message displayed: 0

Inner loop continues **i** = 2 **j** = 1 2 · 1 = 2
Message displayed: 2

Inner loop continues **i** = 2 **j** = 2 2 · 2 = 4
Message displayed: 4

Section 2: Comparing efficiency

When we're concerned with the efficiency of an algorithm, we look at how many operations occur. A single access in an array isn't a big deal, but if the access is within one or more loops, then that statement will be executed *n* times (if the loop goes from 0 to *n-1*)

So if we have a simple loop like this:

```
for ( int i = 0; i < 10; i++ )
{
  // Do a thing
}
```

It will loop *10* times.

And when we have nested for-loops:

```
for ( int i = 0; i < 4; i++ )
{
  for ( int j = 0; j < 3; j++ )
  {
    // Do a thing
  }
}
```

The loop will end up happening *4 x 3* times, or 12 times.

2. For the given code, write down the amount of cycles that occur.

a.

```
for ( int i = 0; i < 100; i++ )
{
   arr[i] += 2;
}
```

(___/2)

Cycles: 100

b.

```
for ( int i = 0; i < 5; i++ )
{
   arr[i] = 0;
}
for ( int i = 5; i < 10; i++ )
{
   arr[i] = 1;
}
```

(___/2)

Cycles: 5 + 5 = 10

c.

```
for ( int i = 0; i < 5; i++ )
{
   for ( int j = 0; j < 3; j++ )
   {
      arr[i] = j;
   }
}
```

(___/2)

Cycles: 5 · 3 = 15

d.

```
for ( int i = 0; i < 5; i++ )
{
   for ( int j = i; j < 5; j++ )
   {
      arr[i] = j * 2;
   }
}
```

(___/2)

Cycles: 15

i	j = i	times
0	0	5
1	1	4
2	2	3
3	3	2
4	4	1
	total	15

e. (___/2)

```
for ( int x = 0; x < 3; x++ )
{
   for ( int y = 0; y < 5; y++ )
   {
      for ( int z = 0; z < 7; z++ )
      {
         arr[x] = y * z;
      }
   }
}
```

Cycles: $3 \cdot 5 \cdot 7 = 105$

f. (___/2)

```
for ( int x = 0; x < 10; x++ )
{
   for ( int y = x+1; y < 10; y++ )
   {
      for ( int z = y+1; z < 10; z++ )
      {
         arr[x] = y * z;
      }
   }
}
```

Cycles:

Summations

$\sum_{i=0}^{n=10} i$ 10 (for loop)

$\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$

$= \frac{10(10-1)}{2} = \frac{90}{2} = 45$ (for second loop)

Combinations

n = number of indices (10)

l = number of loops

2 loops $C(10,2)$

$n = 10$

$l = 2$

$\frac{n!}{(n-l)!\,l!} \rightarrow \frac{10!}{(10-2)!\,2!} = \frac{10 \times 9}{2 \times 1} = \frac{90}{2} = 45$ ✓

3 loops $C(10,3)$

$n = 10$

$l = 3$

$\frac{10!}{(10-3)!\,3!} = \frac{10 \times 9 \times 8}{3 \times 2 \times 1}$

$= \frac{720}{6}$

$= 120$

120 cycles