CSCE 210 QUIZ 1 (RECURSION)

SOLUTIONS

1. Complete the following recursive method for summing a range of an array $a[L \ldots U]$ and $0 \leq L \leq U$ where U is less than the size of the array. In this case the recursive method should return the sum of all array entries $a[L], a[L+1], \ldots, a[U]$.

You may not modify the part of the method already given.

```
int sum(int [] a, int L, int U)
{
   if (L > U) return 0;
   if (L == U) return a[L];
   // put the rest of the  code here









}
```

SOLUTION:
For the non-base case, recursively computer the sum of the array entries

`a[L+1]`, ..., `a[U-1]`

and then add the two entries at the end of the sub-range, namely `a[L]` and `a[U]`.

```
int sum(int [] a, int L, int U)
{
   if (L > U) return 0;
   if (L == U) return a[L];

   return a[L] + a[U] + sum(a, L+1, U-1) ;
}
```

2. Complete the following recursive method for computing the minimum value of a segment of an array $a[0 \ldots U]$[1]. For example, if the array is

array values:	[	34,	23,	12,	78,	100]
array indices:		0	1	2	3	4

and the call is `minimum(a, 1)` then the method returns 12. However, if the call is `minimum(a, 3)` then the call returns 78.

```
int minimum( int [] a, int U)
{
  // put the rest of the code here



}
```

SOLUTION:
Unfortunately this question has a typographical error. The question should have asked for computing the minimum value of of a segment $a[L \ldots a.\text{length} - 1]$ of the array, instead of $a[0 \ldots U]$.

As a result of the typo, the question was deleted from the quiz and everybody got 5 free points.

Try and solve the corrected problem. The method would be

```
int minimum( int [] a, int L)
{
  // put the rest of the code here









}
```

[1]This should have been $a[L \ldots a.\text{length} - 1]$.

3. Complete the following recursive method for printing the segment of a string from a given index to the end of the string. For example, the call

```
printstring("NorthCentral", 3);
```

will print "`thCentral`" while the call `printstring("North", 5);` will print the empty string.

HINT: you can use `System.out.print(str.charAt[k])` to print the single character at position `k` of the string `str`.

```
void printstring(String str, int L)
{
   // place your code here

}
```

SOLUTION:
The base case is when the parameter `L` is equal to the length of the string: in this case there is nothing to do. So we only have something to do for the non-base case, which is when `L < str.length()`.

```
// print the part of the string from
// L until the end of the string.

void printstring(String str, int L)
{
   if (L < str.length())
   {
       // print the character at L
       System.out.print(str.charAt[L]);

       // recursively print the part of the string
       // from L+1 until the end.
       printString(str, L+1);
   }
}
```

4. In the Quicksort algorithm, we make use of a method called

```
int partition(int [] arr, int lower, int upper)
```

which returns the index of an element called the *pivot*. Suppose the method returns an integer `p` where the value of `arr[p]` is `X`.

(a) What can you say about the values `arr[k]` when `lower` $\leq$ `k` $<$ `p`?

(b) What can you say about the values `arr[k]` when `p+1` $\leq$ `k` $<$ `upper`?

SOLUTION:

(a) `arr[k] < X`.

Notice that saying that these values are to the left of `p`, or saying that they are between `lower` and `p`, is just repeating or rephrasing the information

`lower` $\leq$ `k` $<$ `p`

which is already given in the question.

The correct answer avoids confusing `p` and `arr[p]`. The former is an index position into the array, the latter is the value of the array at index position `p` and is equal to `X`.

Similar considerations apply in part (b).

(b) `arr[k] >= X`.

5. Assuming that you have a method `partition` that works as described in Question 4, complete the following code for the recursive Quicksort method:

```
void Quicksort(int [] arr, int L, int U)
{


}
```

SOLUTION:

```
void Quicksort(int [] arr, int L, int U)
{
    if (L < U)
    {
       int p = partition(arr, L, U);
       Quicksort(arr, L, p-1);
       Quicksort(arr, p+1, U);
    }
}
```

Some of the mistakes made were:

(1) Forgetting the base case.
(2) Putting the call to
```
int p = partition(arr, L, U);
```
outside of the base case.
(3) Including type information in a method call, as in
```
Quicksort(int [] arr, int  L, int p-1);
```
Type information belongs in a method definition, never in a method call.
(4) Not knowing whether to use `p`, `p+1`, or `p-1` in the recursive call to Quicksort.
(5) Not studying the lecture notes on Quicksort before the quiz.