Christopher Wilcox
Assignment 2
CS368 Spring 2010
Due: 2010/02/10

1. What will the contents of array be after the execution of this code fragment? Briefly explain your answer.

```
        int *p, a[5] = {1,2,3,4,5};

        p = a;
        p++;
        p[3] = 0;
```

ANSWER:
When you increment the pointer, you will be looking at a[1] as the head of p. the array a = {1,2,3,4,0}

2. Is there any error(s) in the following program? If so, identify each error, and categorize it as a compile time error or a run time error.

```
     #include <iostream>

     int main(void) {
        int a[5]={1,2,3,4,5};
        int *b;
        int i = 0;

        while ( i < 5 ) {
           b[i] = a[i] * a[i]; //This will access areas outside of the array a... Could be considered a runtime error
           b++;
           a++;               //arrays cannot be incremented. Compile Time Error
           i++;
        }

        for (i=0; i < 5; i++)
           cout << a[i];   //cout not declared. Compile time

        return 0;
     }
```

ANSWER:
The above code has a few issues. b[i] will go out of range, as the instruction b++ is advancing the start of b through the a array. Also, a is an array, and cannot be simply incremented. This has no meaning for an array.

3. Does the following code compile without errors? If yes, what value will be printed? If no, explain why not.

```
#include <iostream>
//missing using namespace std;

int main(void) {
   int a[5] = {1,2,3,4,5};
   cout << a[5];  //a[5] is out of the bounds of the array a
   return 0;
}
```

ANSWER:
The above code has some issues... First, It needs the line 'using namespace std;' if you wish to use cout. Even once you add this line though, your output will still be unpredictable, as the element at index 5 is out of the array a, as the array a is only length 5... The last element of array a is at a[4], not a[5]...

4. Assume that we have a singly linked list like the one presented in the class lecture notes. This list differs in that it holds characters, instead of integers.

```
struct Node {
  char value;
  node *next;
};

Node  *front = NULL;  // Pointer to first element in list.
                      // Initialized to NULL to indicate empty list.
```

* Complete the following code fragment to add a first node to this linked list. Use the character 'B'.

ANSWER:

```
// create node for character 'B'
Node bNode = { 'B', NULL };


// add node to currently empty list
if ( front == NULL ) {
   // add to linked list
   front = &bNode;
}
```

* Draw a diagram of what the linked list looks like if there are 3 items in this linked list: 'B', 'C', and 'D'.

ANSWER:

```
NAME (VALUE, NEXT)
bNode('B', cNode)
  ||
  \/
cNode('C', dNode)
  ||
  \/
dNode('D', NULL)
```

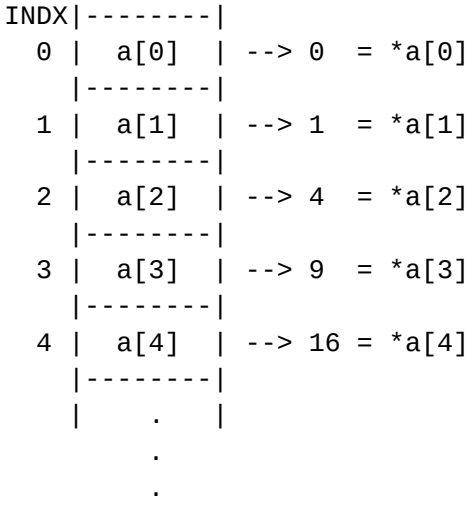
* Would it be easier (and more efficient) to add new items to the beginning (front) or to the end of this linked list? Briefly, explain your answer.

ANSWER: Adding to the front is easier and more efficient... We would need to take what is in the node field of front, place it in our new element, and put our new element as the next node. To put our items at the end, we need to find the end, as it would constantly be changing... One way around this, would be to use a currNode pointer to watch the end of the list for you... that would make adding to the end easier.

5. Suppose we have an array of 25 elements. Each element is a pointer to an integer. Assume that the pointers and integers are already set to initial values.

* Draw a diagram of (at least the first part of) the array, if each integer value is the square of the array index.

ANSWER:
Inside the array, you have a pointer, each pointer points to an integer value... if you dereference the pointer you can get the information it contains (the square of the array index)



* Write a code fragment (a loop) that prints out each integer, without using the square brackets ([]) operator.

ANSWER:

```
int i;
for(i=0; i<25; i++){
    cout << *(a+1);
}
```