CSC 209 Review 5 Solution

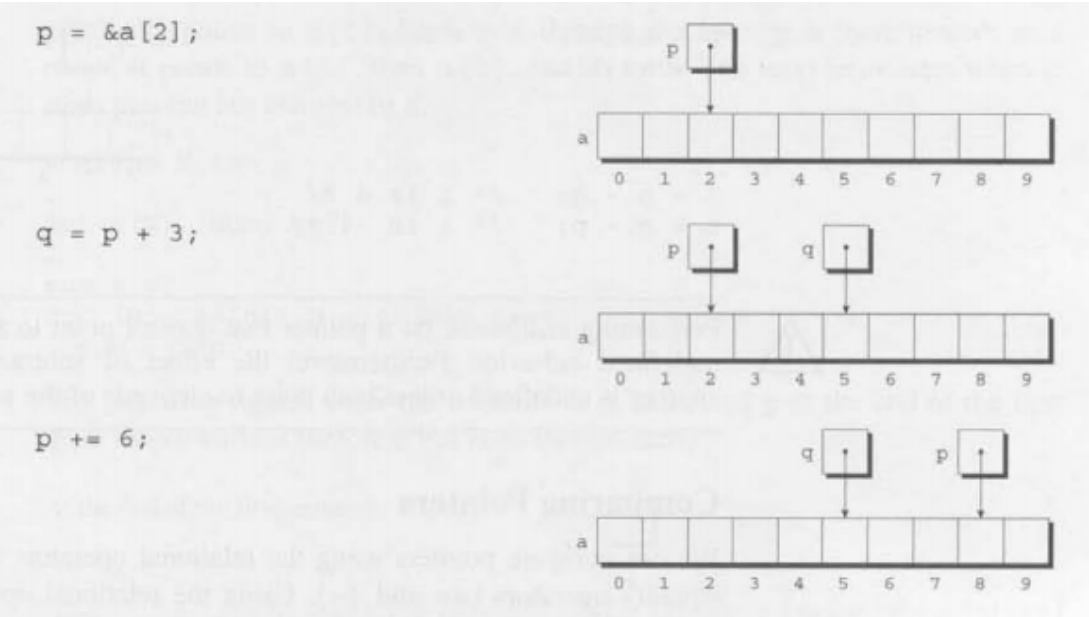
August 21, 2020

1. a) 14
   b) 34
   c) 4
   d) true
   e) false

   Notes

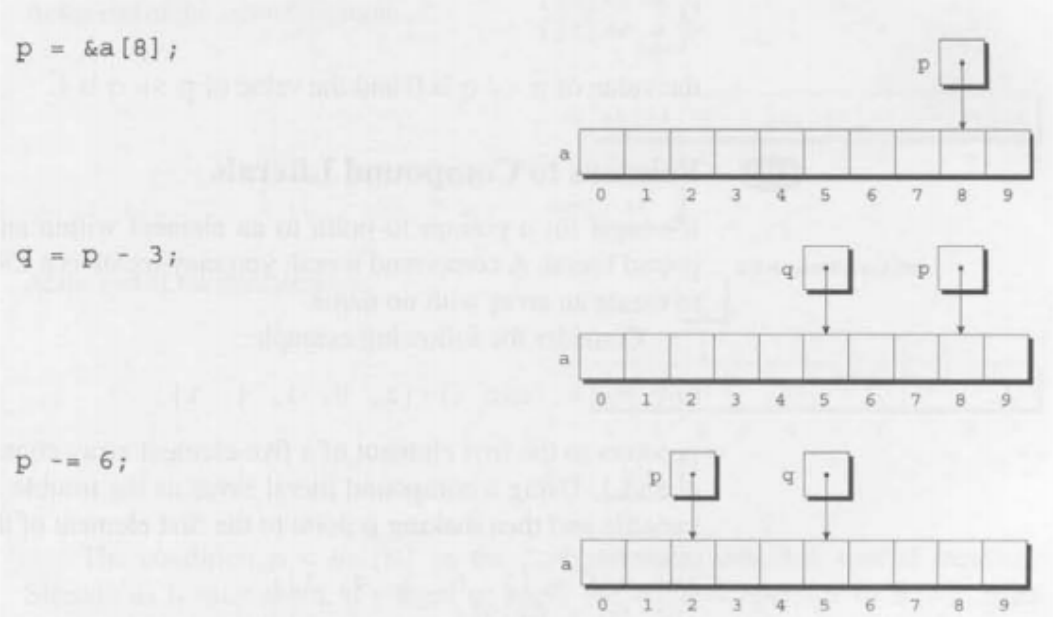
   - **Pointer Arithematic**
     - Adding an integer to a pointer

       Example



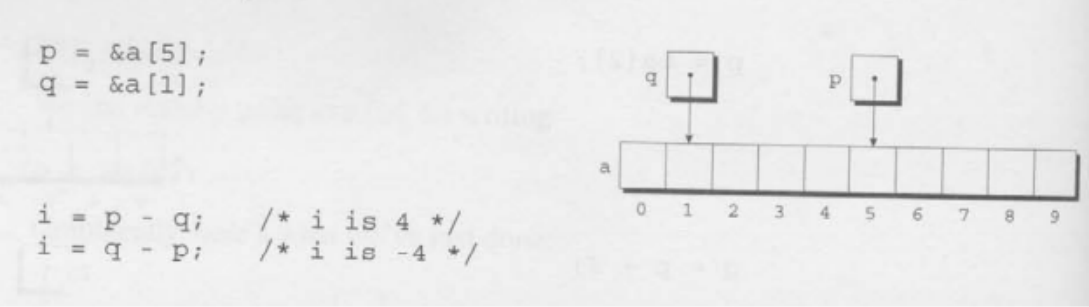
– Subtracting an integer from a pointer

Example



– Subtracting one pointer from another

Example



- **Comparing pointers**
  - Can compare pointers using relational operators (i.e. $<, <=, >, >=$) and the equality operators (i.e. $==, !=$)
  - Returns 1 if `true` and 0 if `false`

Example

```
p = &a[5];
q = &a[1];
```

`p` <= `q` is 0 and `p` >= `q` is 1

2. `low` and `high` are memory addresses.

   So, `low + high` is out of bound, and it could potentially point to an undesirable or wrong value.

   To fix this, we subtract the from high value to the low value:

$$\texttt{middle} = \frac{\texttt{low + high}}{2} \tag{1}$$

3. I need to write the contents of an array `a` after the execution of statements outlined in problem sheet.

   After execution, the array would have contents of $[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$.

   Notes

   - **Combining the * and ++ Operators**
     - `*p++` or `*p++` → Value of expression is `*p` before increment; increment `p` later
     - `(*p)++` → Value of expression is `*p` before increment; increment `*p` later
     - `*++p` or `*(++p)` → Increment `p` first; value of expression is `*p` after increment
     - `++*p` or `++(*p)` → Increment `*p` first; value of expression is `*p` after increment

   Example

   `a[i++] = j`

   Means assign the value `j` to `a[i]` before increment

   Example 2

```
for (p = &a[0]; p < &a[N]; p++)
  sum += *p;
```

   Is the same as

```
p = &a[0];
while (p < &a[N])
  sum += *p++;
```

4. I need to re-write prototype `make_empty`, `is_empty` and `is_full` of the following code to use the pointer variable `top_ptr` instead of the integer variable `top`.

```
#include <stdbool.h>

#define STACK_SIZE 100

/*external variables*/
int contents[STACK_SIZE]
int top = 0;

void make_empty(void) {
    top = 0;
}

bool is_empty(void) {
    return top == 0;
}

bool is_full(void) {
    return top == STACK_SIZE;
}
```

And after re-write using `top_ptr` instead of `top` have:

```
#include <stdbool.h>

#define STACK_SIZE 100

/*external variables*/
int contents[STACK_SIZE]
int *top_ptr = &contents[0];

void make_empty(void) {
    top_ptr = &contents[0];
}

bool is_empty(void) {
    return top_ptr == &contents[0];
}

bool is_full(void) {
    return top_ptr == &contents[STACK_SIZE-1];
}
```