## SAQ1

Because they are dynamically growing, their size cannot be determined until we run the program

## SAQ2

On my computer, it only returns 13, because The printf() function returns the number of characters that are printed.

## SAQ3

Advantage is that we can save RAM resource, disadvantage is that when a process references a page not in the resident set than a page fault occurs, which makes the kernel go grab that page from the swap area and load it into RAM. This takes some time.

## SAQ4

In **foo** function, if **var** > 3, the function will define a local variable **value** and return the address of **value**. However, because the local variable is allocated on the stack and when function exist the scope of local variable value is end. So the value of **value** that **ptr** point to cannot guarantee what it is.
If **var** <= 3, the function will return a pointer **ptr** that point to nowhere.

## SAQ5

1. p=i;
   Invalid, because the type of **i** and **p** are different, one is int, the other is a pointer.

2. p = &q;
   Invalid, because the **&q** is the address of a pointer, but p is an address of

3. p = *q;
   Invalid, because ***q** is the an int and **p** is a pointer

4. *p = q;
   Invalid, same reason in #3

5. p = q;
   valid, now **p** and **q** point to the same int variable.

6. p = *&q;
   Invalid, ***&q** is a int, p is a pointer, same reason in 1#

7. p = &*q;

Valid, **&*q** is a pointer point to int, **q** is also a pointer point to int.

8. &p = q;
   Invalid, same reason in #2

9. *p = *q;
   Valid, let the value that **p** point to be the value that **q** point to.

10. &p = &q
    Invalid, we cannot change the address of a pointer.

## SAQ6

1. Static variable has to be initialized when it was declared.
2. **strlen** function return the length of string without counting NULL, but **strcpy** copy including the terminating null character, so the size need to be **strlen(p)+1**