Name: Login: IN-CLASS EXERCISE

Address syntax 1

For this exercise, assume: `sizeof(int)==4 && sizeof(char)==1 && sizeof(void*)==8`

Initializing new variables

```
// c1 is a char initialized to 55 ('7') with an integer literal
char c1 = 55;
```
sizeof(c2) == 1

```
// c2 is a char initialized to 53 ('5') with an integer literal
char c2 = 53;
```
sizeof(c2) == 1

```
// s1 is the address of the first char in a string stored in the data
// segment: "75"
```
sizeof(s1) ==

```
// s2 is an array of char (a string) stored on the stack and initialized
// to "75" using a string literal.
```
sizeof(s2) ==

```
// s3 is an array of char (a string) stored on the stack and initialized
// to "75" using an array initializer containing character literals.
```
sizeof(s3) ==

```
// s4 is an array of char (a string) stored on the stack and initialized
// to "75" using an array initializer containing integer literals.
```
sizeof(s4) ==

```
// s5 is the address of c1.
```
Output:

```
// a_s5 is the address of s5.
```
Output:

Copy from page 1

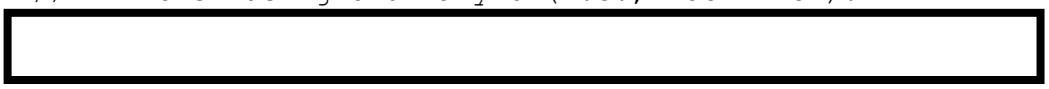
```
// s4 is an array of char (a string) stored on the stack and initialized
// to "75" using an array initializer containing integer literals.
```

```
// s5 is the address of c1.
```

```
// a_s5 is the address of s5.
```

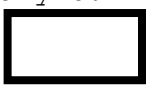
Using addresses in expressions

```
// Print s4 using ordinary C (i.e., not mintf).
```

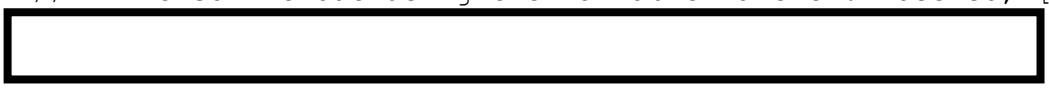


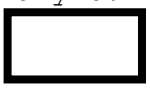
Output: 

```
// Print s5 without using the variable name c1.  Use s5, *, and ordinary C.
```

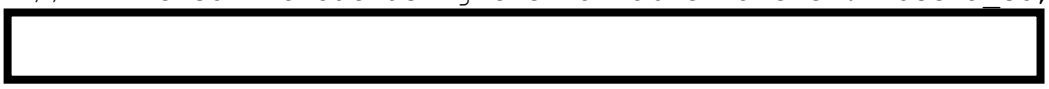
Output: 

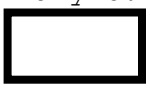
```
// Print s5 without using the variable name c1.  Use s5, […] and ordinary C.
```



Output: 

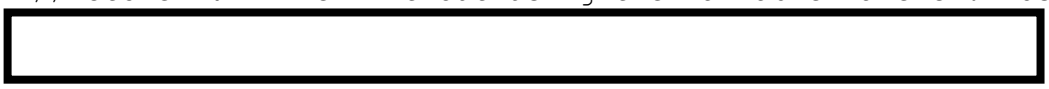
```
// Print s5 without using the variable name c1.  Use a_s5, *, and ordinary C.
```



Output: 

Assignments

```
// Store '?' in c1 without using the variable name c1.  Use s5 and *.
```



sizeof(s5) == 

```
// Store '@' in c1 without using the variable name c1.  Use s5 and […].
```

sizeof(s5) == 

```
// s5 gets the address of c2.
```

sizeof(s5) == 

```
// s5 gets the address of the last character in s4.
```

sizeof(s5) ==