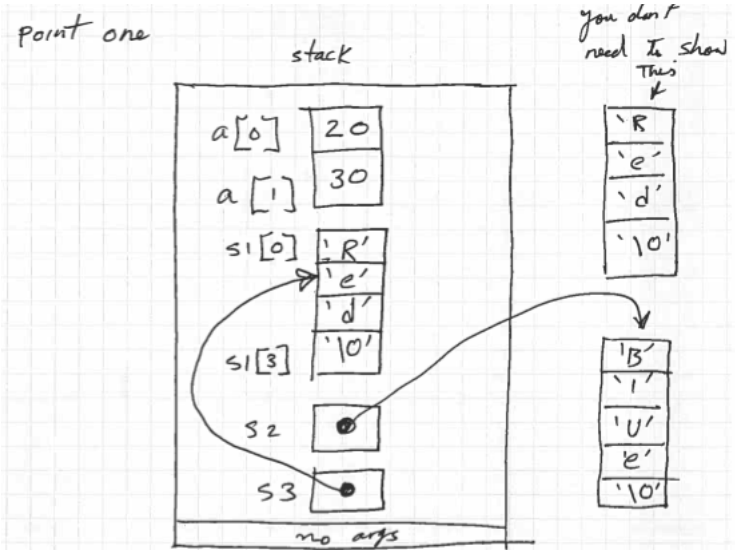
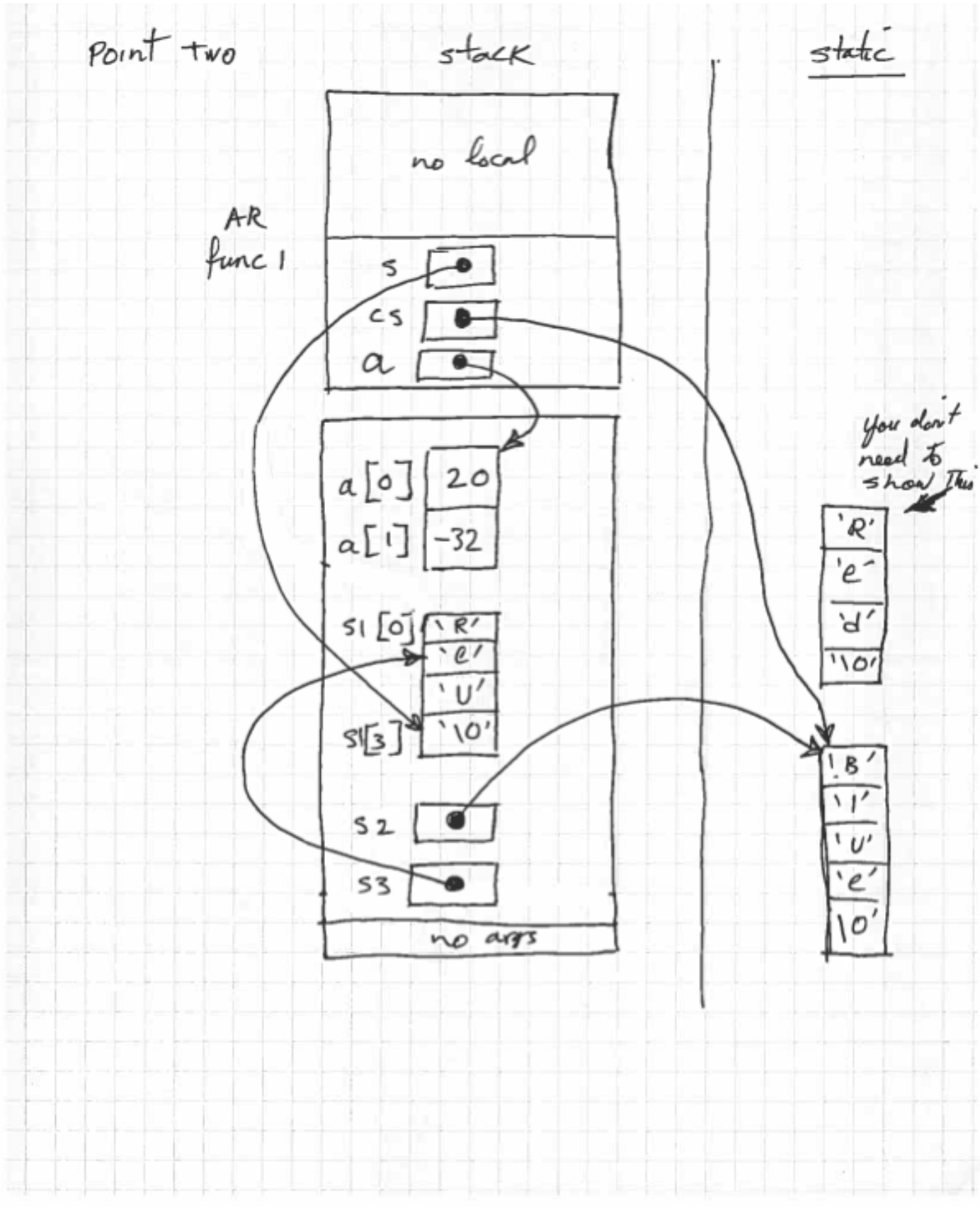
ENSF 337 – Tutorial 3 Fall 2018

Problem I: draw an AR diagram for point one, and two:

```
#include <stdio.h>

void func1(char str1[], const char str2[], int array[]);

int main(){
    int a[]={20, 30};
    char s1[] = "Red";
    const char* s2 = "Blue";
    char * s3;

    s3 = s1+1;
    // Point one

    func1(s1 + 3, s2, a);

    return 0;
}


void func1(char s[], const char cs[], int a[10]){
    *(s-1) = *(cs + 2);
    *(a + 1) = *cs - 'b';

    // Point two
}
```




Point two
stack
static
no local
AR
func 1
s
cs
a
a[0]
20
a[1]
-32
s1[0]
'R'
'e'
'U'
s1[3]
'\0'
s2
s3
no args
You don't need to show this
'R'
'e'
'R'
'\0'
'B'
'I'
'U'
'e'
'\0'

Problem II: What is the output of the following program:

```
#include <stdio.h>

int main(void){
    char s [14] = "AXYBZWR123456";
    char *sp = s;

    int* p = (int*) s;

    printf("%c    %c\n", *p, *sp);

    p++;
    sp++;

    printf("%c    %c\n", *p, *sp);

    p += 2;
    sp += 2;
    printf("%c    %c\n", *p, *sp);
    sp += 10;
    int y = (int) (sp -s);
    printf("%d\n", y);

    return 0;
}
```

Program Output:

```
A    A
Z    X
6    B
13
```

Problem III: In the following space write the definition of a function called my_strcmp that works almost similar to strcmp library function. Returns 0 if the two string are lexicographically are identical, returns -1 one if the first string is smaller than second string, and returns 1 if the first string is greater than second string:
Examples:

```
Returns 0 if s1 is "ABC" and s2 is "ABC"
Returns -1 if s1 is "ABC" and s2 is "BBC"
Returns 1 if s1 is "BBC" and s2 is "ABC"
Returns 0 if s1 is "" and s2 is ""
Returns 1 is s1"aBC" and s2 is "ABC"
Returns -1 is s1 is "AB" and s2 is "ABC"
```

You are not allowed to use square brackets (means you have to use pointer arithmetic) and you are not allowed to call strcmp.

```
int my_strcmp(char *s1, const char* s2);

int main() {
    const char* s1 = "";
    char s2[10] = "1";
    printf("%d\n", strcmp(s2, s1));    // prints  Apple
    printf("%d\n", my_strcmp(s2, s1)); // expected to print: Apple
    return 0;
}
```

```
int my_strcmp(char *s1, const char* s2)
{
    int i=0;
    while (s1[i] != '\0' || s2[i] !='\0')
    {
        if(*s1 > *s2)
            return 1;
        else if(*s1 < *s2)
            return -1;
        s1++;
        s2++;

    }
    return 0;
}
```