CS107
Autumn 2018

Cynthia Lee
November 2, 2018

CS107 Midterm Examination SOLUTIONS

Problem 1: Integer Representation

(a) 0110 1101
(b) -54
(c) 1110 0001
(d) (any)

Problem 2: Pointers and Arrays

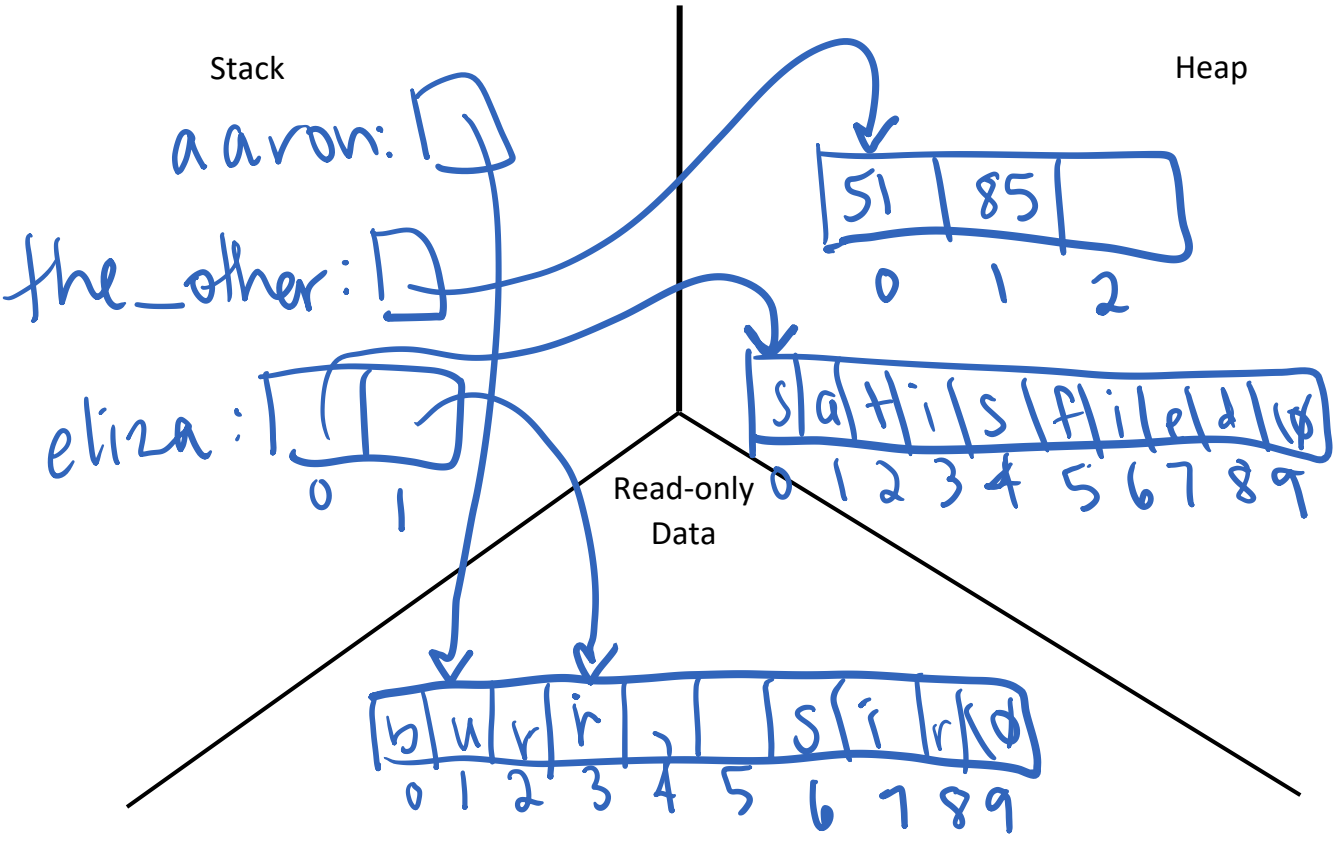
(a) Leave blank or `assert(nelems > 0)`. Although sort of harmless, `assert(nelems >= 0)` is not helpful because its size is unsigned anyway. Code should not be mallocing temp space.

(b)
```
nelems * sizeof(int)
```

(c)
```
copy[i] = *(arr[i]);
free(arr[i]);
arr[i] = copy + i;
```

(d) Leave blank.

Problem 3: Memory Diagram



Text description of the diagram above:

aaron is a char * on the stack, which points to the index 1 element (the 'u') of an array of 10 characters in the read-only data segment (a string literal) containing the characters "burr, sir" and a null terminator. In this data segment array, the character 'b' is at index 0, 'u' at index 1, 'r' at index 2, 'r' at index 3, ',' at index 4, ' ' at index 5, 's' at index 6, 'i' at index 7, 'r' at index 8, and '\0' at index 9.

the_other is an int * on the stack, which points to the index 0 element of an array of 3 ints on the heap. The index 0 element of this array is 51, and the index 1 element is 85. The index 2 element is not initialized.

eliza is an array of 2 char *s on the stack. The index 0 element of this array points to to the index 0 element (the 's') of an array of 10 characters on the heap storing a string "satisfied" and a null terminator. In this heap-allocated string, the character 's' is at index 0, 'a' at index 1, 't' at index 2, 'i' at index 3, 's' at index 4, 'f' at index 5, 'i' at index 6, 'e' at index 7, 'd' at index 8, and '\0' at index 9. The index 1 element of **eliza** points to the index 3 character (the second 'r' in "burr") in the string literal in the data segment also pointed to by **aaron**.

Problem 4: Generics and Function Pointers

(a)

```
void remove_less (void *arr, size_t *nelems, size_t width,
                  int (*cmp)(const void *p, const void *q))
{
    // this guards against nelems = 0
    for (size_t i = (*nelems) ? (*nelems - 1) : 0; i > 0; i--) {
        void *ith = (char*)arr + i * width;
        int res = cmp(ith, arr);
        if (res < 0) {
            memmove(ith, (char*)ith + width, (*nelems - 1 - i) * width);
            *nelems = *nelems - 1; // *nelems--; doesn't work due to op precedence
        }
    }
}
```

(b)

```
int farm_compare(const void *p, const void *q)
{
    const struct farm *farm_p = (const struct farm *)p;
    const struct farm *farm_q = (const struct farm *)q;
    return farm_p->count + strlen(farm_p->species) -
           (farm_q->count + strlen(farm_q->species));
}
```

LAST (FAMILY) NAME: _____ SUNET: _____

Problem 5: Bitwise Operations

(a)

```
bool zeros_detector_loop(unsigned int n)
{
    unsigned int mask = 0x3; // 0b000....00011
    for (int i = 0; i < 31; i++) {
        if (!(n & mask)) return true;
        mask <<= 1;
    }
    return false;
}
```

(b)

```
// one elegant solution
bool zeros_detector(unsigned int n)
{
    return (~n) & (~n << 1);
}
// a alternate mask-based solution
bool zeros_detector(unsigned int n)
{
    return ((n | (n >> 1)) & 0x7FFFFFFF) != 0x7FFFFFFF;
}
```