



Unit of Study Survey

Please take a few moments to complete the end of semester survey:

<http://sydney.edu.au/itl/surveys/complete>

Exercise 1: Memory, `sizeof` and pointers

1. Assume a 64 bit system, where the `sizeof` all pointers are 8 bytes, and `sizeof(int)` is 4 byte. The following questions refer to the code below:

```
void f(void) {
    struct int_array {
        int* data;
        int len;
    };

    struct int_array arr;
    struct int_array* arr_ptr = &arr;
    struct int_array arrs[100];
}
```

- (a) What are the initial values of `arr.data` and `arr.len`
 - (b) What is `sizeof(struct int_array)`?
 - (c) What is `sizeof(arr_ptr)`?
 - (d) What is `sizeof(arr_ptr->data)`?
 - (e) What is `sizeof(arrs)`?
 - (f) Suppose the address of `arrs[0]` is `0x1000`, what is the address of `arrs[50]`?
 - (g) Suppose the address of `arrs[50]` is `0x1500`, what is the address of `arrs[20]`?
 - (h) Suppose the address of `arrs[50]->len` is `0x1500`, what is the address of `arrs[50]->data`?
2. Assume a 64 bit system, where the `sizeof(int)` is 4 bytes, and `sizeof(float)` is also 4 bytes. The following questions refer to the code below:

```
void* g(int x) {
    union merged {
        int x;
        float y;
    };
}
```

```

static int counter = 0;
++count;

union merged* ptr = malloc(sizeof(merged));
if (!ptr) {
    return NULL;
}
ptr->x = x;
return ptr;
}

```

- What is the `sizeof(union merged)`?
- Which variables are allocated on the stack, which variables are allocated on the heap and which variables are allocated in the static storage area?
- What are the lifetimes of each of the variables listed above?

Exercise 2: C preprocessor

- What is wrong with the following macro of a MAX function?

```
#define MAX(x, y) x > y ? x : y
```

- What are include guards?

Exercise 3: Declarations, definitions and linkage

Classify each of the following as a declaration or definition. What are their linkages?

```

int* x;      declare
static int* y;  declare

const int z = 0;  definition
extern const int w;  declare

static void f(void);  declare
static void f(void) {  definition
    return 0;
};

void g(void);  declare
void h(void) {  definition
    return;
};

int main(void) {
    return 1;
}

```

Exercise 4: Control flow

Trace the following program by hand. What do they output?

1. `#include <stdio.h>`

```
int main(void) {
    for (unsigned i = 5; i-- > 0;) {
        printf("%u\n", i);
        if (i) {
            for (int j = 0; j < i; ++j) {
                printf("%d\n", i + j);
            }
        }
    }
}
```

2. `#include <stdio.h>`

```
void cpy2(char* dest, char* src, size_t n) {
    size_t i = 0;
    for (; i < n; ++i) {
        dest[i] = src[i];
    }
}

int main(void) {
    // why does this segfault when x is declared as a char*?
    char x[] = "123456";
    cpy2(x + 1, x, 4);
    puts(x);
    return 0;
}
```

Exercise 5: Strings

Implement the following functions from the `string.h`:

```
void memcpy(void* dest, const void* src, size_t len);
void memmove(void* dest, const void* src, size_t len);
int memcmp(const void* s1, const void* s2, size_t len);

void strcpy(char* dest, const char* src);
void strcat(char* dest, const char* src);
char* strchr(const char* s, int c);
char* strrchr(const char* s, int c);
size_t strspn(const char* s, const char* accept);
```

Exercise 6: Parallelism and concurrency

1. Why is `x += 1` not an atomic operation?
2. What are the four conditions of a dead lock?
3. What is a critical section?
4. What is a live lock?
one or more threads in your program are blocked from gaining access to a resource and, as a result, cannot make progress.
5. What is starvation in term of multi-threading?
6. What does it mean if a resource is highly contended, why could this cause a performance issue?
7. What is false sharing?
8. How can a binary semaphore be used as a mutex?
9. The Senate Bus problem from the Little Book of Semaphores

“This problem was originally based on the Senate bus at Wellesley College. Riders come to a bus stop and wait for a bus. When the bus arrives, all the waiting riders invoke `boardBus`, but anyone who arrives while the bus is boarding has to wait for the next bus. The capacity of the bus is 50 people; if there are more than 50 people waiting, some will have to wait for the next bus. When all the waiting riders have boarded, the bus can invoke `depart`.

If the bus arrives when there are no riders, it should depart immediately.

Puzzle: Write synchronization code that enforces all of these constraints. ”

Represent each rider as a thread, write synchronisation code to model the above problem.