Common Bugs

Classic Memory Bugs

- Memory management is one of the biggest differences between C and Java
- Let's go over some bugs that might afflict you

Bug #1: scanf() bug

- What is the issue with the following code?

```
scanf(“%d”, val);
```

- scanf() stores input using pointer arguments
- Need to use an ampersand (&) to address variable val

```
scanf(“%d”, &val);
```

Bug #2: Memory Allocation And Use

- What is the issue with the following code?

```
/* return y = Ax */
int *matvec(int **A, int *x) {
    int *y = (int *) malloc(N*sizeof(int));
    int i, j;

    for (i=0; i<N; i++)
        for (j=0; j<N; j++)
            y[i] += A[i][j]*x[j];

    return y;
}
```

- Assumes that heap data is initialized to zero

Bug #3: Overwriting Memory

- What is the issue with the following code?

```
int **p;
p = (int **)malloc(N*sizeof(int));

for (i=0; i<N; i++) {
    p[i] = (int*)malloc(M*sizeof(int));
}
```

- Allocating the possibly wrong sized object

Bug #4:

- What is the issue with the following code?

```
int **p;
p = (int **)malloc(N*sizeof(int *));
for (i=0; i<=N; i++) {
   p[i] = (int *)malloc(M*sizeof(int));
}
```

- Off by one error, goes past array bounds

Bug #5: Pointers

- What is the issue with the following code?

```
int *search(int *p, int val) {
    while (*p && *p != val)
        p += sizeof(int);
    return p;
}
```

- Misunderstanding pointer arithmetic

Bug #6

- What is the issue with the following code?

```
int *foo () {
    int val;
    return &val;
}
```

- Forgetting that local variables disappear when a function returns

Bug #7

- What is the issue with the following code?

```
x = malloc(N*sizeof(int));
...
free(x);
y = malloc(M*sizeof(int));
...
free(x);
```

- Freeing the same memory multiple times

Bug #8

- What is the issue with the following code?

```
x = malloc(N*sizeof(int));
...
free(x);
...
y = malloc(M*sizeof(int));
for (i=0; i<M; i++)
    y[i] = x[i]++;
```

- Referencing freed memory

Bug #9

- What is the issue with the following code?

```
void foo() {
    int *x = malloc(N*sizeof(int));
    ...
    return;
}
```

- Failing to free dynamically allocated memory
  - Memory leak
  - Will slowly eat up memory over time

Bug #10

- What is the issue with the following code?

```
struct list {
    int val;
    struct list *next;
};

void foo() {
    struct list *head = (struct list*) malloc(sizeof(struct list));
    head->val = 0;
    head->next = NULL;
    <create and manipulate the rest of the list>
    ...
    free(head);
    return;
}
```

- Freeing only part of a data structure