Solutions to the Worksheet

1. Memory Allocation Table

Code Fragment	Space	Where?	De-allocated When?
`int main() { int i; }`	`sizeof(int)`	Stack frame for `main`	When program ends
`int fun() { float i; }`	`sizeof(float)`	Stack frame for `fun`	When `fun()` returns
`int main() { fun(); }`	None	N/A	N/A
`int fun(char i) { ... }`	`sizeof(char)`	Stack frame for `fun`	When `fun()` returns
`int main() { fun('a'); }`	`sizeof(char)` for `fun('a')`	Stack frame for `fun`	When `fun()` returns
`int main() { char i[10] = {'h','o','i'}; }`	`sizeof(char) * 10`	Stack frame for `main`	When program ends
`int main() { char *i; }`	`sizeof(char *)`	Stack frame for `main`	When program ends
`int main() { int *i; }`	`sizeof(int *)`	Stack frame for `main`	When program ends
`int fun(int *i) { ... }`	`sizeof(int *)`	Stack frame for `fun`	When `fun()` returns
`int main() { int i[5] = {4,5,2,5,1}; fun(i); }`	`sizeof(int) * 5`	Stack frame for `main`	When program ends
`int main() { int *i; i = malloc(sizeof(int)); }`	`sizeof(int *)` for `i`, `sizeof(int)` for `malloc`	`i` in stack, allocated memory in heap	`i` de-allocated at program end, heap memory with `free()`
`void fun(int **i) { *i = malloc(sizeof(int) * 7); }`	`sizeof(int *)` in `fun`, `sizeof(int) * 7` in heap	Stack frame for `fun`, allocated memory in heap	Stack frame cleared when `fun()` returns, heap memory de-allocated with `free()`
`int main() { int *i; fun(&i); free(i); }`	`sizeof(int *)` for `i`, `sizeof(int) * 7` in heap	`i` in stack, allocated memory in heap	`i` de-allocated when program ends, heap memory with `free()`

2. Memory Tracing for malloc Example

Heap Memory (Dynamic Allocation):

- Address `0x23c`: `numbers2[0]` = 0

- Address `0x240`: `numbers2[1]` = 1
- Address `0x244`: `numbers2[2]` = 2

Stack Frame for `initialize`:

- Address `0x454`: `a1[0]` = 0
- Address `0x458`: `a1[1]` = 1
- Address `0x45c`: `a1[2]` = 2
- Address `0x460`: `a2[0]` = 0
- Address `0x464`: `a2[1]` = 1
- Address `0x468`: `a2[2]` = 2

Stack Frame for `main`:

- Address `0x474`: `numbers1[0]` = 0
- Address `0x478`: `numbers1[1]` = 1
- Address `0x47c`: `numbers1[2]` = 2
- Address `0x480`: `numbers2[0]` = 0
- Address `0x484`: `numbers2[1]` = 1
- Address `0x488`: `numbers2[2]` = 2