COMP1511 Tutorial 8

malloc | linked lists

malloc

Allocates memory that persists outside local scope i.e. won't be deleted when a function ends.

```
void *malloc(size_t size)
```

Takes the amount of memory to be allocated and returns the address of the allocated memory.

free

If we don’t free memory when we’re done, the computer will never be able to use this memory for anything else - this is a memory leak

If we cause too many memory leaks the computer can eventually run out of memory

We can check for memory leaks with leakcheck:

```
dcc --leakcheck -o filename c_file
```

free

Usage: `void free(void *pointer)`

E.g. `free(head);`

(where head is a pointer to the memory we want to free)

Remember: we are freeing memory, not freeing pointers!

Visualising Linked Lists

```
// Step 1:
// malloc memory for a new node called node1

// Step 2:
// node1 data = 3
// node1 next = NULL

// Step 3:
// make the head pointer points to node1

// Step 4:
// malloc memory for a new node called node2

// Step 5:
// node2 data = 9
// node2 next = NULL

// Step 6:
// add node2 to the tail of the list, making node1 next point to node2

// Step 7:
// malloc memory for a new node called node3

// Step 8:
// node3 data = 5
// node3 next = NULL

// Step 9:
// add node3 to the head of the list, making node3 next point to the current
// head of the list
// make the head point to node3
```

```c
struct node {
    int data;
    struct node *next;
};
```


int data = 1
struct node
*next = 0xFF40
int data = 3
struct node
*next = 0xFFA0
int data = 5
struct node
*next = NULL
NULL
0xFF00
0xFF40
0xFFA0