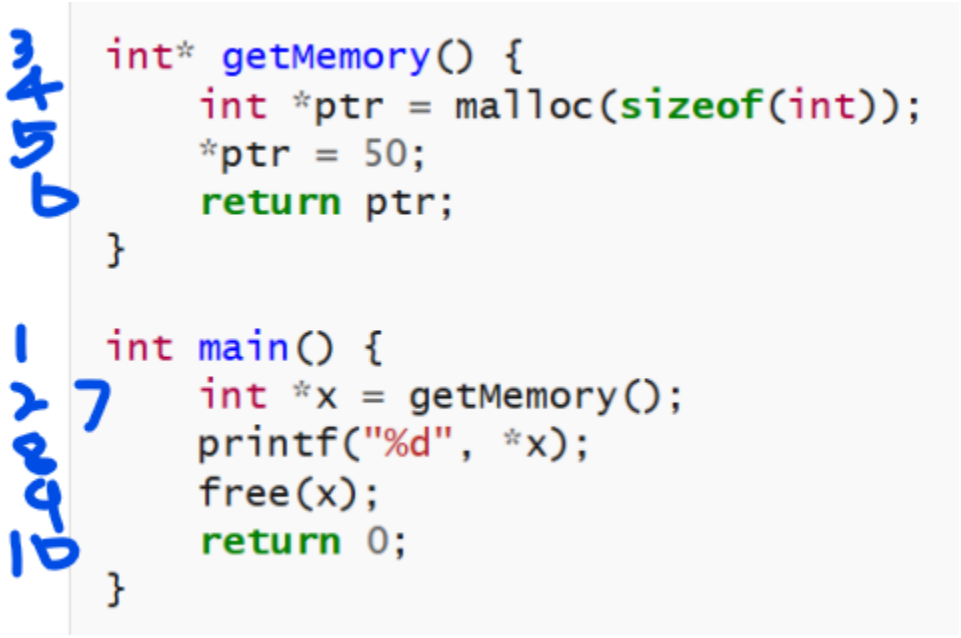
Siddarth Pai

Assignment - C Programming

1. **Output is 20 or some garbage value(undefined)**
   The output given by the machine is undefined and can vary from machine to machine as once you use free to deallocate the memory, it becomes a dangling pointer. The pointer will remember the address but the right to store memory in it is no longer ours. So depending on the machine it will either store 20 or some garbage value. Output might be 20 since the pointer still points to that address, so when I do *p = 20, it will store 20 in the address that p points to.
2. **Output is 50**



```
int* getMemory() {
    int *ptr = malloc(sizeof(int));
    *ptr = 50;
    return ptr;
}

int main() {
    int *x = getMemory();
    printf("%d", *x);
    free(x);
    return 0;
}
```

3. **Output is 0**
   Unlike Malloc, Calloc assigns an initial value of 0 to all allocated memory spaces
4. **Output is 129, a**
   ptr points to address of a
   &cho is just referencing and is equal to the variable ch only(in address and value)
   cho = ‘A’+32 = 97 = ‘a’
   ptr = 32+97 = 129
   Since ptr affects a and ch affects cho and vice versa, output is 129, a
5. The code gives compile error because we are trying to do the following:

a) Assigning ptr to be constant and then trying to modify it by doing ++
b) Similar to a but we assign ptr to address of j which is again invalid since its supposedly a constant

```
#include <iostream>
using namespace std;
int main()
{
const int i = 20;
const int* const ptr = &i;
(*ptr)++;
int j = 15;
ptr = &j;
cout << i;
return 0;
}
```

6.

Output is cprog

Static makes the array global variable and it can be called between function call statements now

In the flow of program,

str = rtos

arr = rtos

str = arr

str = arr = cprog

cprog