Array notation	Equivalent pointer notation
a[i]	*(a + i)
a[0]	*a
&a[i]	a + i
a[i] = v	*(a+i) = v
a[0] = v	*a = v

```
void* allocate_at(uint64_t* start, size_t size) {
  size_t current_size = block_size(start[0]);
  if(current_size > size) {
    uint64_t remaining = current_size - size - SLOT_SIZE;
    int next_block_index = (size / SLOT_SIZE) + 1;
    start[next_block_index] = remaining; // even, free
  }
  start[0] = size | 1; // busy
  return &start[1];
}
```

```
void* allocate_at(uint64_t* start, size_t size) {

  size_t current_size = block_size(________________);


  if(current_size > size) {
    uint64_t remaining = current_size - size - SLOT_SIZE;
    int next_block_index = (size / SLOT_SIZE) + 1;


   ____________________= remaining; // even, free
  }
  start[0] = size | 1;


  return  __________________;
}
```

Q1: Fill in the blanks to use pointer notation

```
void free(void* ptr) {
  uint64_t* slot_after_header = ptr;
  slot_after_header[-1] = slot_after_header[-1] - 1;
}
```

```
void free(void* ptr) {
  uint64_t* slot_after_header = ptr;


  ________________________________________

}
```

Q2 do the same in free()

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>

int main() {
  int value = 17;
  int pid = fork();
  if(pid == 0) {
    printf("In child before assignment: %p => %d\n", &value, value);
    value = 999;
    while(1) {
      printf("In child:\t%p => %d\n", &value, value);
    }
  }
  else {
    printf("In parent before assignment: %p => %d\n", &value, value);
    value = 3333;
    while(1) {
      printf("In parent:\t%p => %d\n", &value, value);
    }
  }
}
```