Exercise 4

Garbage Collection

Write a copying GC

- Given:
  - Objects:
    - Lists (Pointer to a list: bit 0 set. L=cons(a,b) => *ht=a, *(ht+1) = b, L = ht | 1, ht+=2),
    - 30-bit integers. (*ht = i << 2).
  - Roots: Linked c-list of pointers, *roots*.
  - start_f_s, end_f_s, start_t_p, end_t_s, ht: (heap_top pointer into f-s).
- Write a copying GC in C.
  - Algorithm:

```
scan = next = start of to-space
for each root r { r = forward(r); }
while scan < next {
        if is_list(*scan) {
        for each field f of *scan
                scan->f = forward(scan->f)
        }
        scan += size(*scan)
}
ht = start_t_s; swap_spaces()
```

Possible solution

```
#define is_list(p) ((p) & 1)
#define size(p) 2
#define is_copied(p) (start_t_s < (*p) &&
    end_t_s > (*p))
uint32 *scan, *next, *start_f_s, *end_f_s,
                 *start_t_s, *end_t_s,
    *ht;

#define swap_spaces() do {\
   uint32 *t1 = start_f_s;       \
   start_f_s = start_t_s          \
   start_t_s = t1;                     \
   uint32 *t2 = end_f_s;         \
   end_f_s = end_t_s;            \
   end_t_s = t2;                       \
} while (0)

uint32 *forward(uint32 *p) {
   if(! is_copied(p)) {
       void *tp = p;
       void *tn = n;
       for(int *i = size(p); i > 0;
           i--, t++, tn++) *tn = *t;
       *p = next;
       next = tn;
   }
   return *p;
}
```

```
typedef struct _p_list {
    uint32 *p;
    struct _p_list *next;
} p_list;

void gc(p_list roots) {
   scan = next = start_t_s;
   while (roots) {
       *roots->p = forward(roots->p);
        roots = roots->next;
   }
   while (scan < next) {
        if (is_list(*scan)) {
             *scan = forward(*scan);
             scan++;
             *scan = forward(*scan);
         }
         scan++;
   }
   ht = start_t_s;
   swap_spaces();
}
```