CASS: Exercise session 6

Linked Lists

Dynamic memory: recap

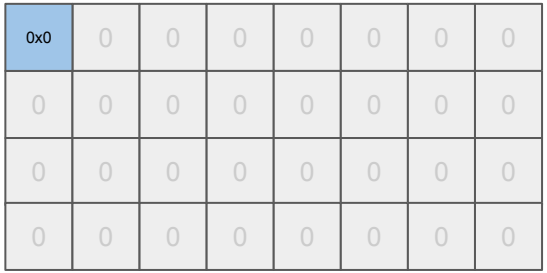
Memory

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Visual

s = stack_create()

Memory



Visual

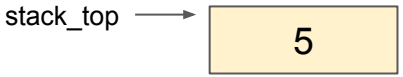
stack_top →

stack_push(s, 5)

Memory

0x1	5	0x0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Visual

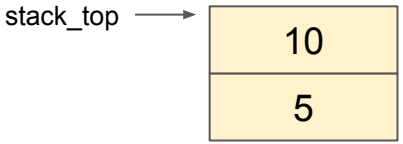


stack_push(s, 10)

Memory

0x3	5	0x0	10	0x1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Visual

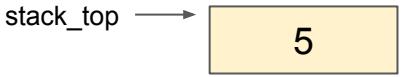


stack_pop(s)

Memory

0x1	5	0x0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Visual

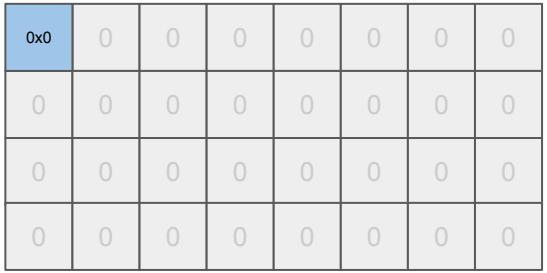


Dynamic memory in C

- Use structs to define data structure
- Use malloc to allocate space on heap
- Use free to free space on heap

s = stack_create()

Memory

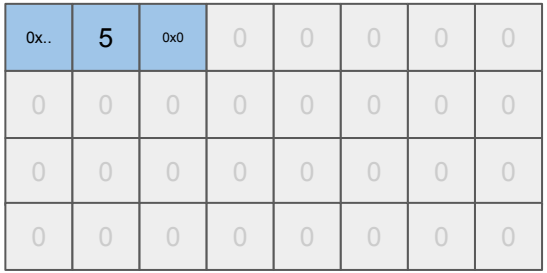


C

```
struct stack {
      struct stack_element *top
}

struct stack_element {
      ...
}

struct stack *create_stack(){
      struct stack *s =
            malloc(sizeof(struct stack));
      s->top = NULL;
      return s;
}
```

stack_push(s, 5)

Memory



C

```
struct stack_element {
      int val;
      struct stack_element *next;
}

void *stack_push(struct stack* s,
                 int val){
      struct stack_element *element =
            malloc(sizeof(struct stack_element));
      element->val = val;
      element->next = stack->top;
      stack->top = element;
}
```

stack_push(s, 10)

Memory

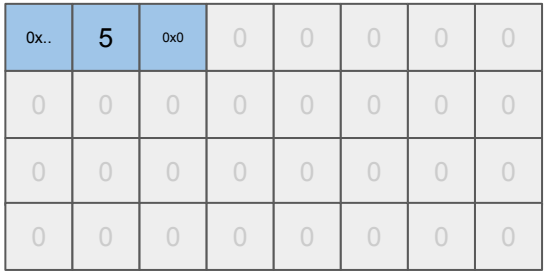
0x..	5	0x0	10	0x..	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

C

```
struct stack_element {
      int val;
      struct stack_element *next;
}

void *stack_push(struct stack* s,
                  int val){
      struct stack_element *element =
            malloc(sizeof(struct stack_element));
      element->val = val;
      element->next = stack->top;
      stack->top = element;
}
```

stack_pop(s)

Memory



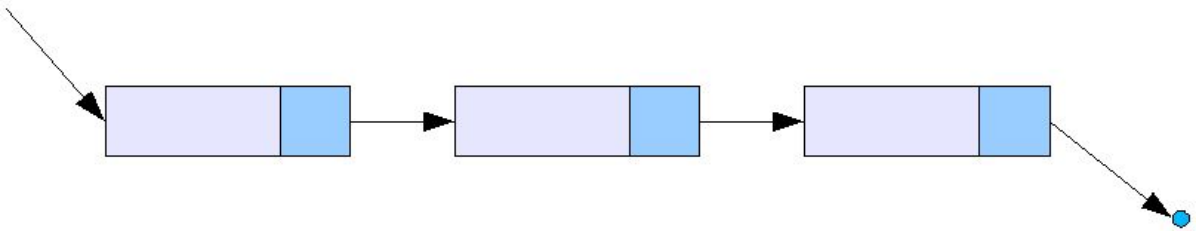
C

```
struct stack_element {
      int val;
      struct stack_element *next;
}

int *stack_pop(struct stack* s){
      struct stack_element *popped =
            stack->top;
      int popval = popped->val;
      s->top = popped->next;
      free(popped);
      return popval;
}
```

Linked lists

- Like lists, linked lists store values (e.g., integers)
- Unlike arrays, where values are stored contiguously in memory, linked lists “link” elements together through pointers
- The last element is indicated by a NULL pointer

```
struct List {
    struct ListElement* first;
}

struct ListElement {
    int key;
    struct ListElement* next;
}
```



Exercise info

- C
  - Use make command to build
    - make && ./linked_list
- RARS
  - Provided with skeleton files
    - Enable assemble all files in this directory!
  - Provided a malloc/free implementation
    - jal malloc, jal free
  - Error message with 0xdeadbeef: caused by test suite!
    - Double click to see failed assertion
    - Also check I/O output for potential hints
  - Adhere to calling conventions, this is tested
    - If jal_and_check fails, didn’t restore callee-save reg!