Additional linked list operations

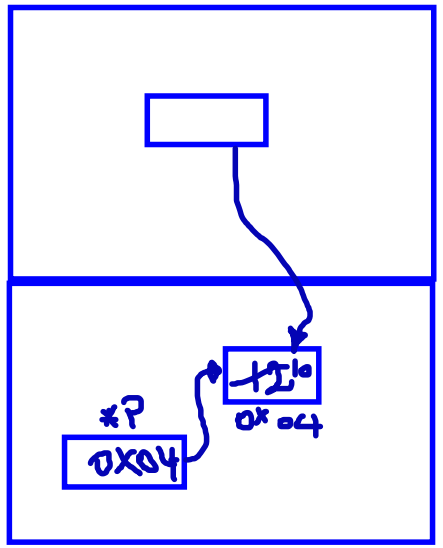
- clear - deallocates all nodes in the list, sets head pointer to null
- add_front
- clear_list (free all nodes)
- remove_after
- remove_front
- remove_all (remove all occurrences of a particular data value)

Pointers are by pass by value

pointer_pv.c:

```c
#include <stdio.h>

void fun1(int * ip) {
     *ip = 10;
     ip += 1; // increment the address
}
int main() {
     int a = 12;
     int * p = &a;
     printf("p points to address %p with value %d\n", (void *)p, *p);
     fun1(p); // pass p by value; changes to p will NOT affect p
     printf("p points to address %p with value %d\n", (void *)p, *p);
     return 0;
}
```




```
$ gcc -std=c99 -pedantic -Wall -Wextra pointer_pv.c
$ ./a.out
p points to address 0x7ffcb5871c64 with value 12
p points to address 0x7ffcb5871c64 with value 10
```

Pass a pointer by reference

pointer_pr.c:

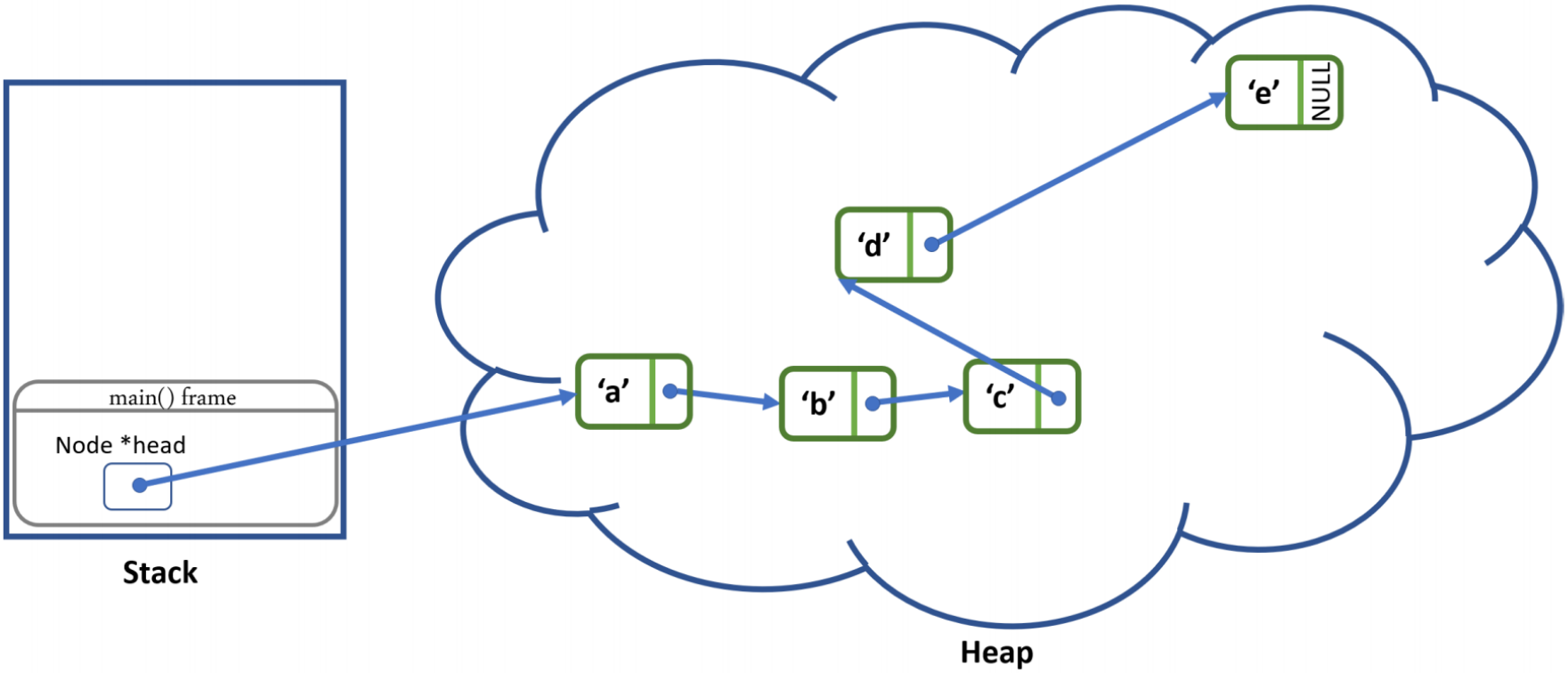
```
#include <stdio.h>

void fun1(int ** ip) {
     *ip += 1; // increment the address
}
int main() {
     int a = 12;
     int * p = &a;
     printf("p points to address %p with value %d\n", (void *)p, *p);
     fun1(&p); // passing p by reference; any changes WILL impact p
     printf("p points to address %p with value %d\n", (void *)p, *p);
     return 0;
}
```

```
$ gcc -std=c99 -pedantic -Wall -Wextra pointer_pr.c
$ ./a.out
p points to address 0x7ffef43d4b0c with value 12
p points to address 0x7ffef43d4b10 with value 4198832
```

Linkedlist head

- The linked list *head* should be passed by reference if it needs to be updated

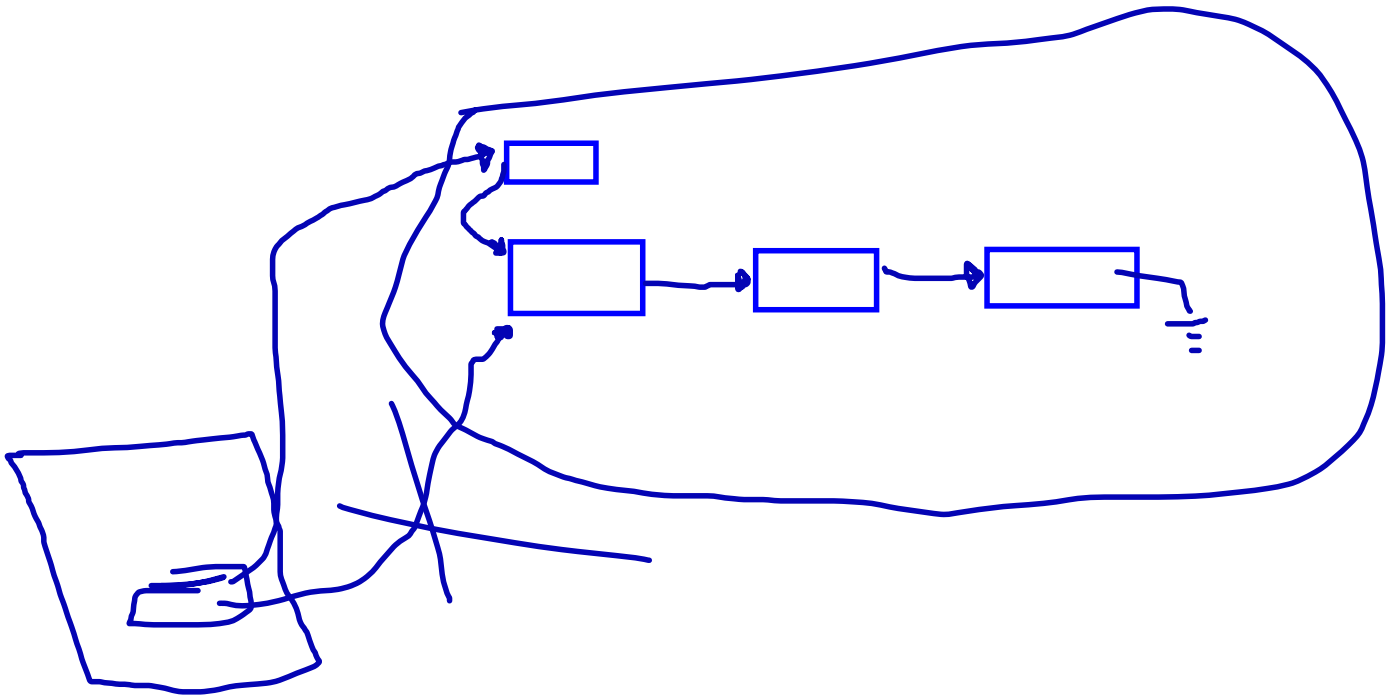


add_after vs. add_front

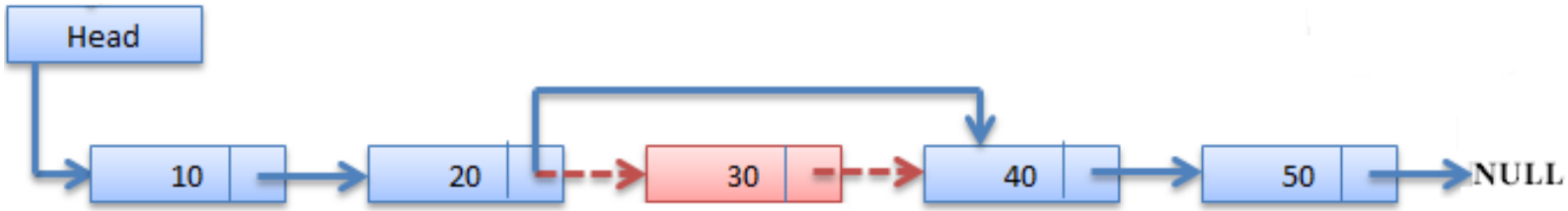
- `void add_after(Node * node, char val);`
- `void add_front(Node ** list_ptr, char val);`
  - needs ability to modify actual head pointer (not a copy), so call with `&head` as argument

Example add_front call: `add_front(&head, value);`

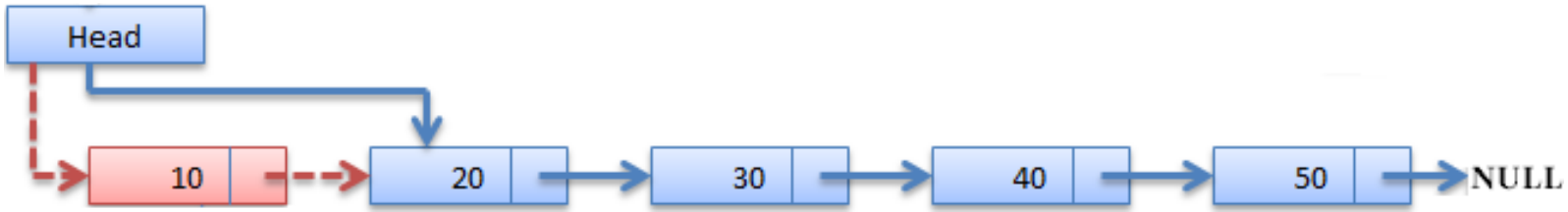
```
void add_front(Node ** list_ptr, char val) {
     Node * n = create_node(val);
     n->next = *list_ptr;  //new node's next gets address of old first node
     *list_ptr = n;  //head pointer gets address of new node
}
```



Delete Operations




```
char delete_after(Node * node);
```




```
char delete_front(Node ** list_ptr);
```

"*list_ptr" to get the address of the first node (i.e., the content of head)