Intermediate Programming

Day 19

Announcements

- If you are not part of the group by tonight, we will assign you to one.

Outline

- Exercise 18

Exercise 18

Implement **remove_after**.

list.c

```
...
char remove_after( Node *node )
{
        Node *n = node->next;
        if( !n ) return '?';
        char data = n->data;
        node->next = node->next->next;
        free( n );
        return data;
}
...
```

Exercise 18

Implement **remove_front**.

list.c

```
...
char remove_front( Node **list_ptr )
{
        Node *n = (*list_ptr);
        if( !n ) return '?';
        char data = n->data;
        *list_ptr = n->next;
        free( n );
        return data;
}
...
```

Exercise 18

Implement **remove_all**.

list.c

```
...
void remove_all( Node **list_ptr , char val )
{
        while( (*list_ptr)->data==val )
                remove_front( list_ptr );
        for( Node *n=*list_ptr ; n ; n=n->next )
                while( n->next && n->next->data==val )
                        remove_after( n );
}
...
```

Exercise 18

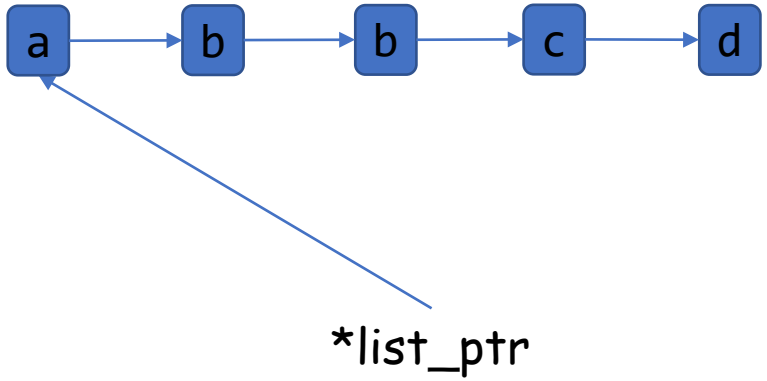
Implement **remove_all**.

Note that it has to be **while**, an **if** statement won't cut it.

Consider trying to remove all occurrences of the character **b** from the list.

list.c

```
…
void remove_all( Node **list_ptr , char val )
{
        while( (*list_ptr)->data==val )
                remove_front( list_ptr );
        for( Node *n=*list_ptr ; n ; n=n->next )
                while( n->next && n->next->data==val )
                        remove_after( n );
}
…
```



Exercise 18

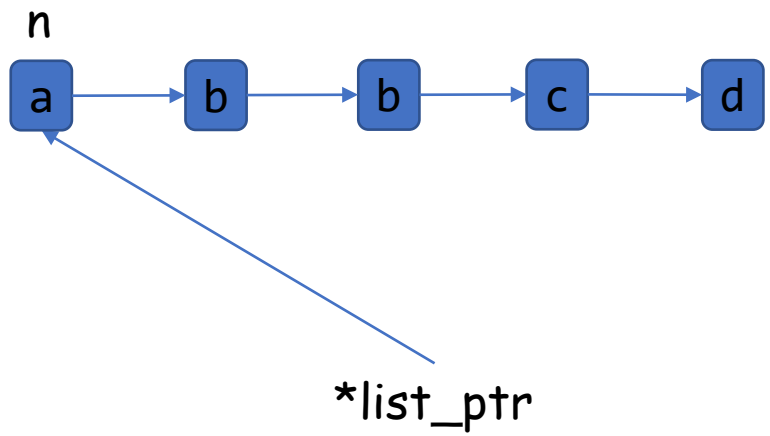
Implement **remove_all**.

Note that it has to be **while**, an **if** statement won't cut it.

Consider trying to remove all occurrences of the character **b** from the list.

list.c

```
…
void remove_all( Node **list_ptr , char val )
{
        if( (*list_ptr)->data==val )
                remove_front( list_ptr );
        for( Node *n=*list_ptr ; n ; n=n->next )
                if( n->next && n->next->data==val )
                        remove_after( n );
}
…
```



Exercise 18

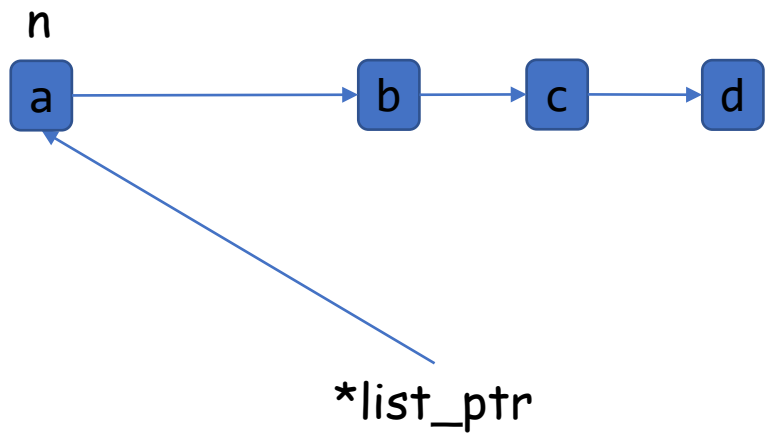
Implement **remove_all**.

Note that it has to be **while**, an **if** statement won't cut it.

Consider trying to remove all occurrences of the character **b** from the list.

list.c

```
…
void remove_all( Node **list_ptr , char val )
{
        if( (*list_ptr)->data==val )
                remove_front( list_ptr );
        for( Node *n=*list_ptr ; n ; n=n->next )
                if( n->next && n->next->data==val )
                        remove_after( n );
}
…
```



Exercise 18

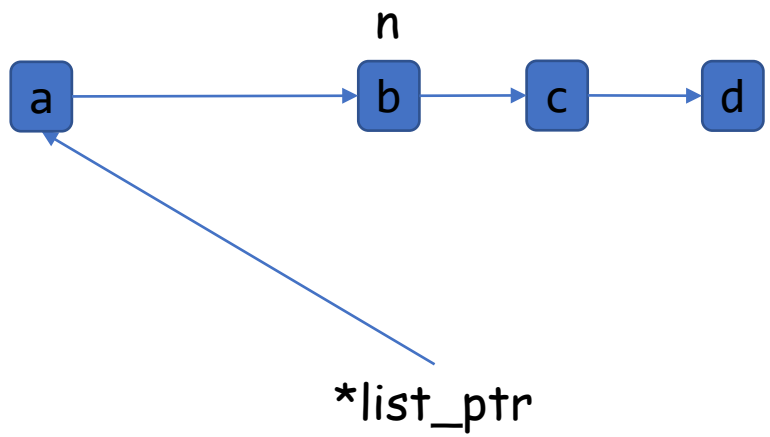
Implement **remove_all**.

Note that it has to be **while**, an **if** statement won't cut it.

Consider trying to remove all occurrences of the character **b** from the list.

list.c

```
…
void remove_all( Node **list_ptr , char val )
{
        if( (*list_ptr)->data==val )
                remove_front( list_ptr );
        for( Node *n=*list_ptr ; n ; n=n->next )
                if( n->next && n->next->data==val )
                        remove_after( n );
}
…
```



Exercise 18

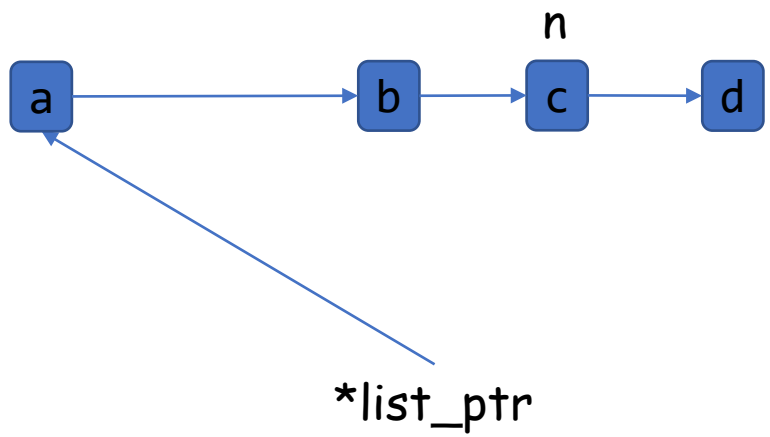
Implement **remove_all**.

Note that it has to be **while**, an **if** statement won't cut it.

Consider trying to remove all occurrences of the character **b** from the list.

list.c

```
…
void remove_all( Node **list_ptr , char val )
{
        if( (*list_ptr)->data==val )
                remove_front( list_ptr );
        for( Node *n=*list_ptr ; n ; n=n->next )
                if( n->next && n->next->data==val )
                        remove_after( n );
}
…
```



Exercise 18

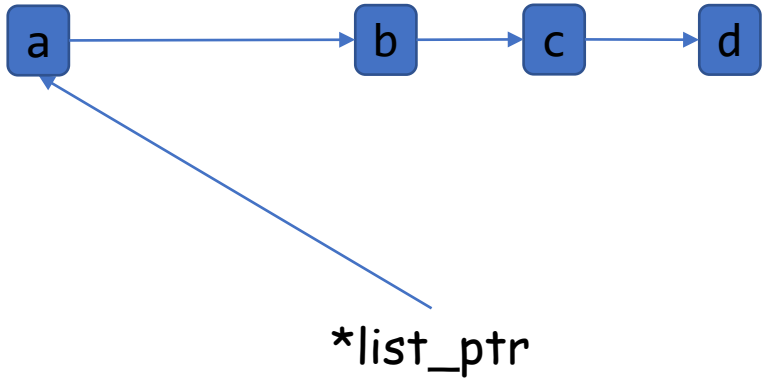
Implement **remove_all**.

Note that it has to be **while**, an **if** statement won't cut it.

Consider trying to remove all occurrences of the character **b** from the list.

list.c

```
…
void remove_all( Node **list_ptr , char val )
{
        if( (*list_ptr)->data==val )
                remove_front( list_ptr );
        for( Node *n=*list_ptr ; n ; n=n->next )
                if( n->next && n->next->data==val )
                        remove_after( n );
}
…
```



Exercise 18

Implement **insert**.

- If the list is empty create the list.
- If **val** comes before the first entry in the list, add it to the front.
- Otherwise find the node in the list that is smaller than **val** but with next node that is bigger.

list.c

```
…
Node *insert( Node **list_ptr , char val )
{
      if( !*list_ptr )
      {
            *list_ptr  = create_node( val );
            return *list_ptr;
      }
      else if( val<(*list_ptr)->data )
      {
            add_front( list_ptr , val );
            return *list_ptr;
      }
      else
      {
            Node *n;
            for( n=*list_ptr ; n->next && val>=n->next->data ; n=n->next ) ;
            add_after( n , val );
            return n->next;
      }
}
…
```