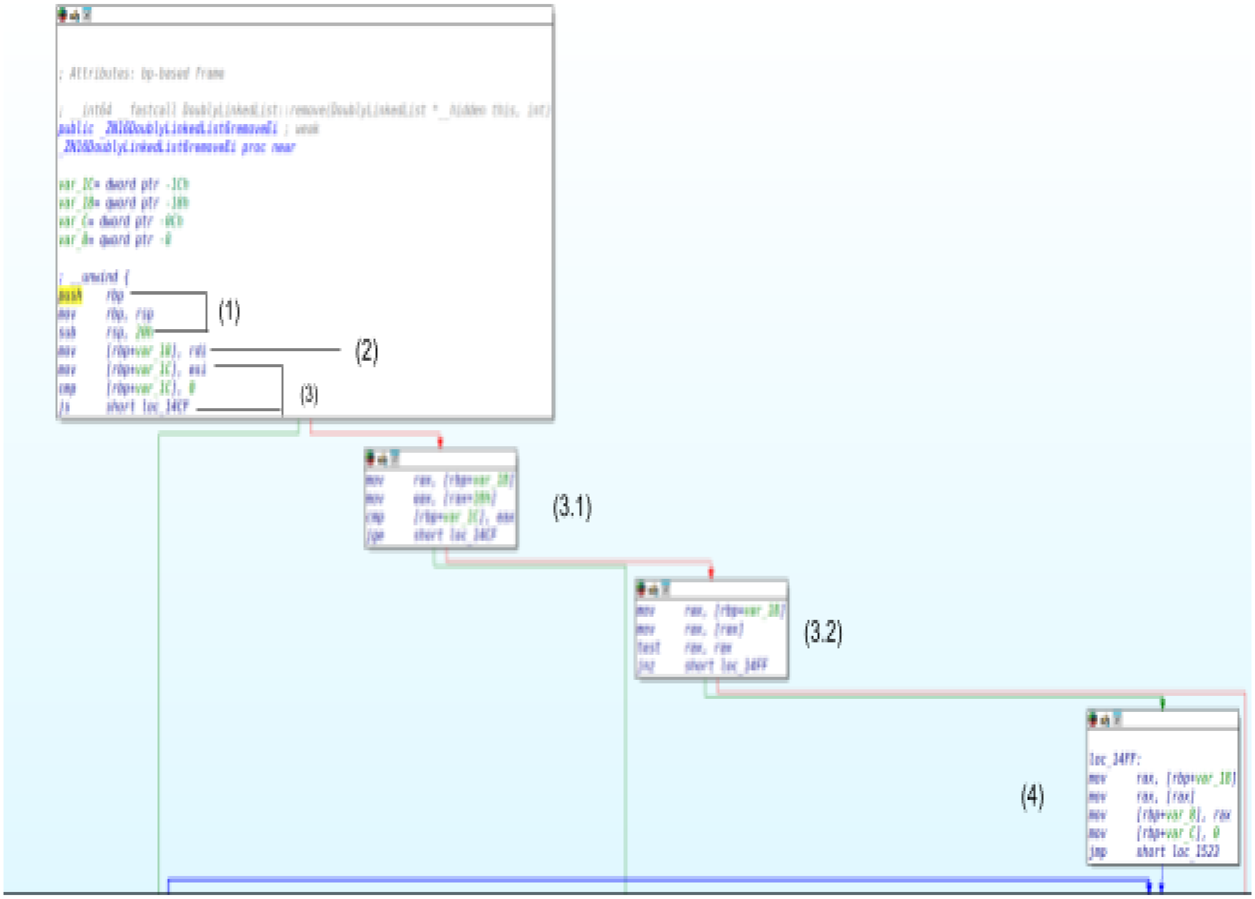


```
; Attributes: bp-based frame

; __int64 __fastcall DoublyLinkedList::remove(DoublyLinkedList *__hidden this, int)
public _ZN16DoublyLinkedList6removeEi ; weak
_ZN16DoublyLinkedList6removeEi proc near

var_1C= dword ptr -1Ch
var_18= qword ptr -18h
var_C= dword ptr -0Ch
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 20h
mov     [rbp+var_18], rdi
mov     [rbp+var_1C], esi
cmp     [rbp+var_1C], 0
js      short loc_14CF
```

```
mov     rax, [rbp+var_18]
mov     eax, [rax+10h]
cmp     [rbp+var_1C], eax
jge     short loc_14CF
```

```
mov     rax, [rbp+var_18]
mov     rax, [rax]
test    rax, rax
jnz     short loc_14FF
```

```
loc_14FF:
mov     rax, [rbp+var_18]
mov     rax, [rax]
mov     [rbp+var_8], rax
mov     [rbp+var_C], 0
jmp     short loc_1523
```

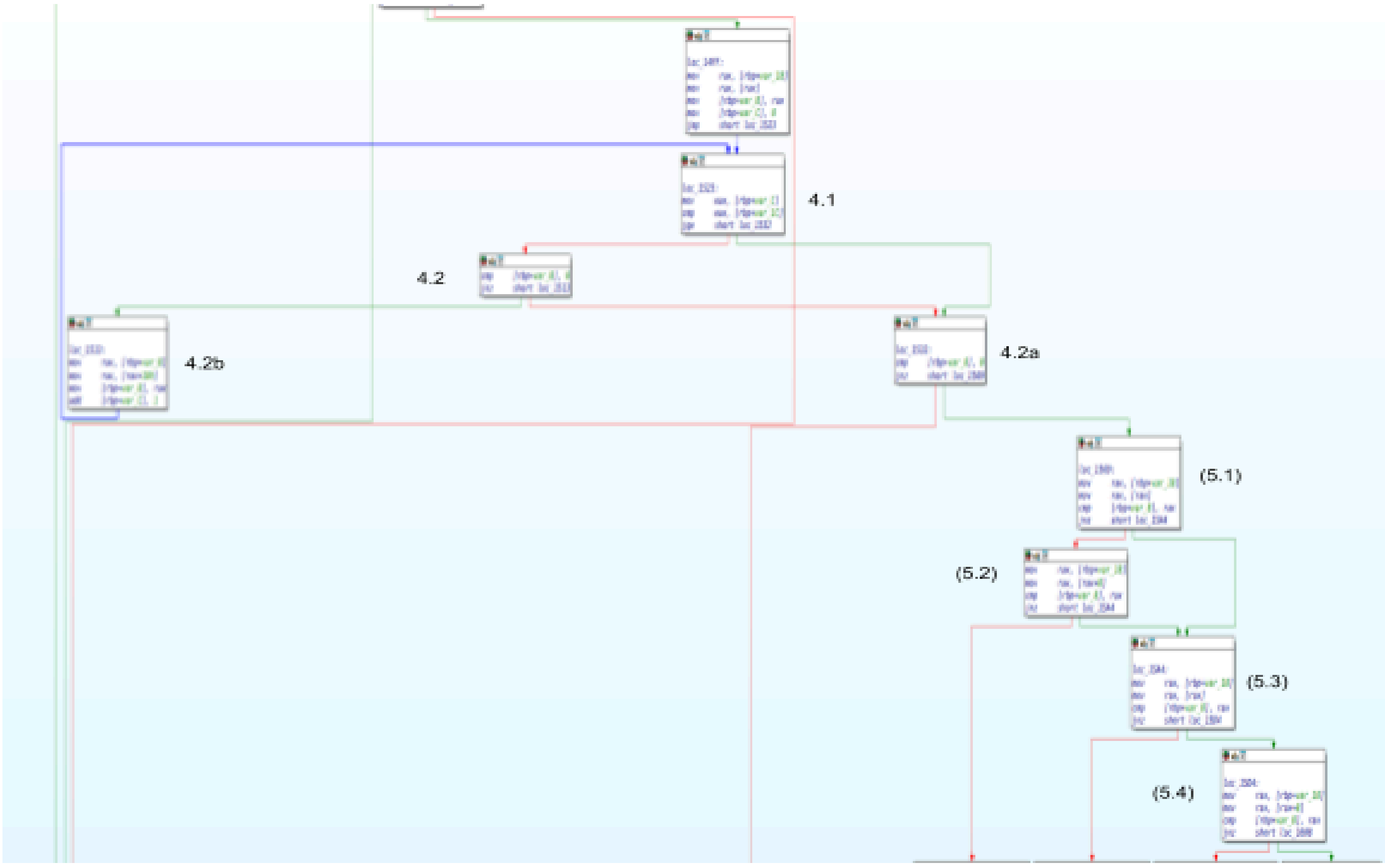
1.Function Preamble

2. Loads in where head is stored in memory for later checking

3. Loads in the given index for the function and checks if it’s less than zero. If so, it goes to the Invalid Index block.

- 3.1
    - Loads in len of the list and compares it to the provided index for an OOB check
- 3.2
    - Loads in the head of the list to see if the list exists in the first place. (empty list)

4.Loop preparation. Sets the current node to the head of the list and i to 0

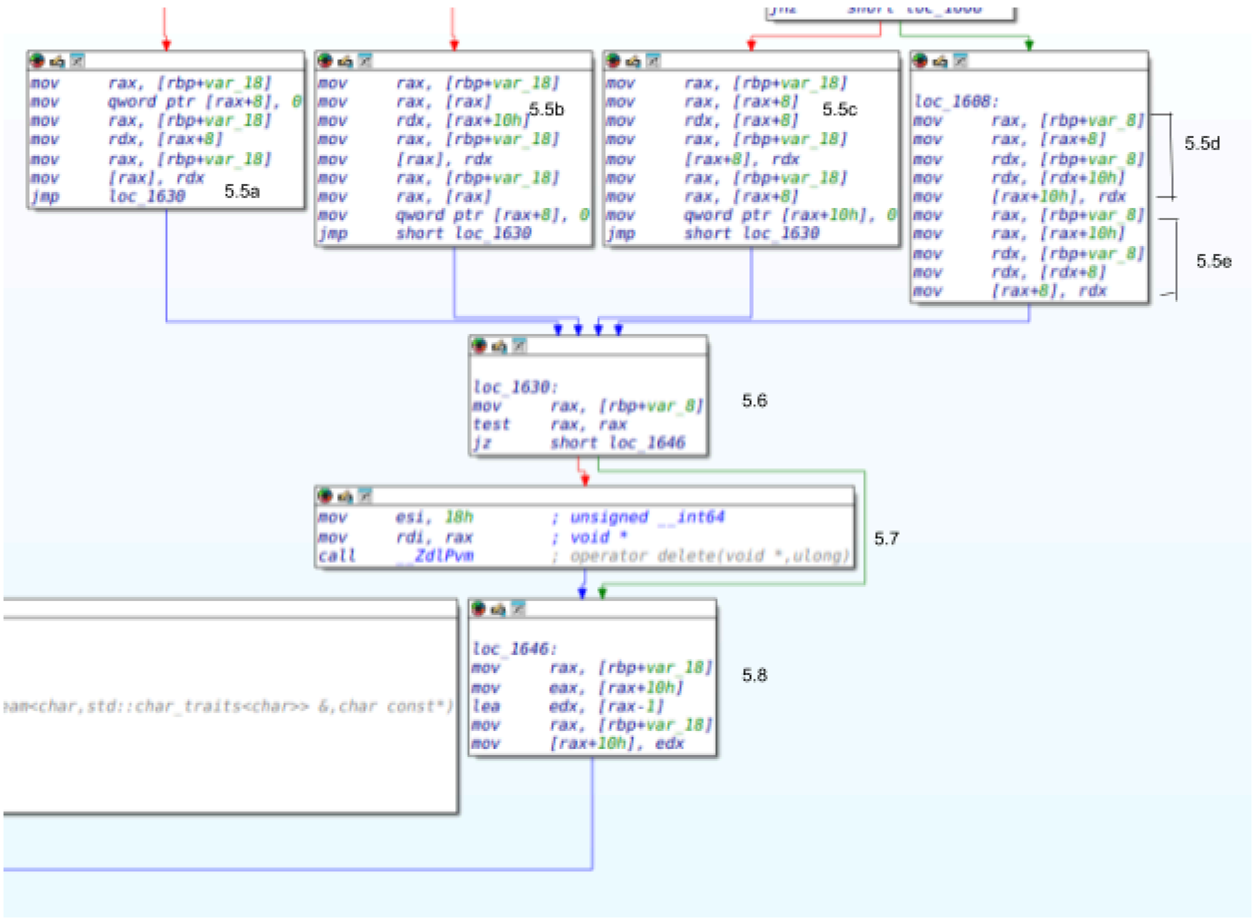


4.

- 4.1
    - Compares the i to the given index to see if the requested indexed node is in the list
- 4.2.
    - 4.2a. Checks if the current node exists and going to the OOB case if it doesnt exist (NOTE: this is handled within the loop as the loop only keeps going if current exists, likely could be optimized out)
    - 4.2b. Increments i by 1 and sets current to current->next

5. These handle the various current node cases, whether curr is equal to the head of the list, the tail of the list, or both.

- 5.1. Checks if current node is head (for both case)
- 5.2. Checks if current node is tail (for both case)
- 5.3. Compares current node to head for its own case
- 5.4. Compares current node to tail in its own if case



```
jnz     short loc_1608
```

5.5a

```
mov     rax, [rbp+var_18]
mov     qword ptr [rax+8], 0
mov     rax, [rbp+var_18]
mov     rdx, [rax+8]
mov     rax, [rbp+var_18]
mov     [rax], rdx
jmp     loc_1630
```

5.5b

```
mov     rax, [rbp+var_18]
mov     rax, [rax]
mov     rdx, [rax+10h]
mov     rax, [rbp+var_18]
mov     [rax], rdx
mov     rax, [rbp+var_18]
mov     rax, [rax]
mov     qword ptr [rax+8], 0
jmp     short loc_1630
```

5.5c

```
mov     rax, [rbp+var_18]
mov     rax, [rax+8]
mov     rdx, [rax+8]
mov     rax, [rbp+var_18]
mov     [rax+8], rdx
mov     rax, [rbp+var_18]
mov     rax, [rax+8]
mov     qword ptr [rax+10h], 0
jmp     short loc_1630
```

5.5d

```
loc_1608:
mov     rax, [rbp+var_8]
mov     rax, [rax+8]
mov     rdx, [rbp+var_8]
mov     rdx, [rdx+10h]
mov     [rax+10h], rdx
```

5.5e

```
mov     rax, [rbp+var_8]
mov     rax, [rax+10h]
mov     rdx, [rbp+var_8]
mov     rdx, [rdx+8]
mov     [rax+8], rdx
```

5.6

```
loc_1630:
mov     rax, [rbp+var_8]
test    rax, rax
jz      short loc_1646
```

5.7

```
mov     esi, 18h            ; unsigned __int64
mov     rdi, rax            ; void *
call    __ZdlPvm            ; operator delete(void *,ulong)
```

```
eam<char,std::char_traits<char>> &,char const*)
```

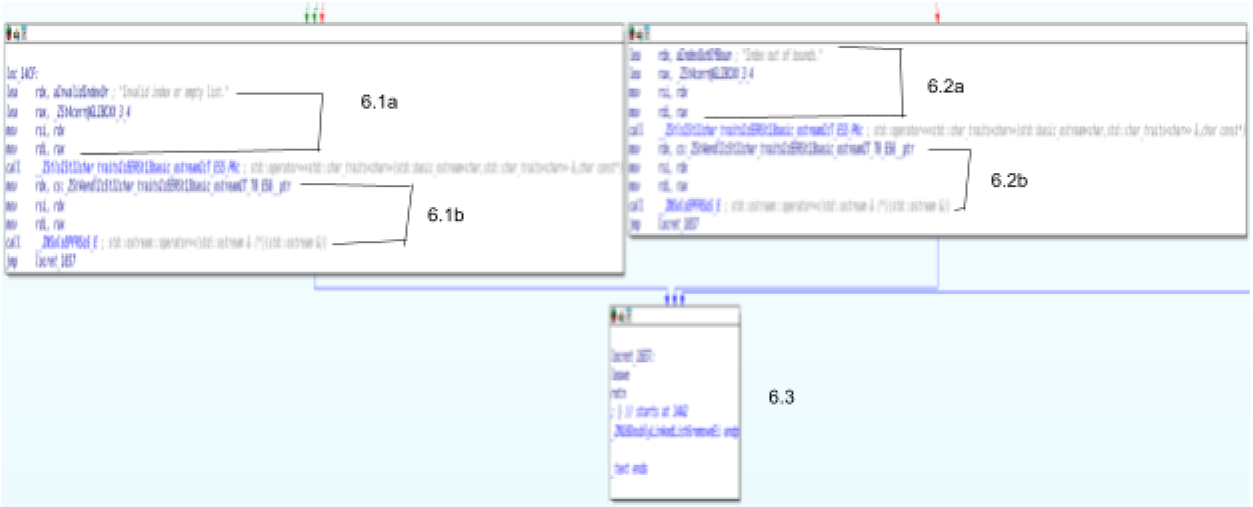
5.8

```
loc_1646:
mov     rax, [rbp+var_18]
mov     eax, [rax+10h]
lea     edx, [rax-1]
mov     rax, [rbp+var_18]
mov     [rax+10h], edx
```

5.5a. Curr = both head and tail, sets nullptr to head and tail
5.5b. Curr = head case, sets head->prev to null and head to head->next
5.5c. Curr = tail case, sets tail->prev to null and tail to tail->next
5.5 d/e.

- 5.5d. Loads in curr->prev->next and sets that equal to curr->next
- 5.5e. Loads in curr->next->prev and sets that equal to curr->prev

5.6. Checks if curr exists before deletion (after loop finishes)
5.7. Delete current node
5.8. Loads in the list and reduces the lists length by 1



```
loc_[illegible]:
lea     rdx, aInvalidIndexOr ; "Invalid index or empty list."
lea     rax, _ZSt4cout@@GLIBCXX_3_4
mov     rsi, rdx
mov     rdi, rax
call    [illegible] ; std::operator<<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,char const*)
mov     rdx, cs:[illegible]_ptr
mov     rsi, rdx
mov     rdi, rax
call    [illegible] ; std::ostream::operator<<(std::ostream & (*)(std::ostream &))
jmp     locret_[illegible]
```

```
lea     rdx, [illegible] ; "Index out of bounds."
lea     rax, _ZSt4cout@@GLIBCXX_3_4
mov     rsi, rdx
mov     rdi, rax
call    [illegible] ; std::operator<<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,char const*)
mov     rdx, cs:[illegible]_ptr
mov     rsi, rdx
mov     rdi, rax
call    [illegible] ; std::ostream::operator<<(std::ostream & (*)(std::ostream &))
jmp     locret_[illegible]
```

```
locret_[illegible]:
leave
retn
; } // starts at [illegible]
[illegible] endp

_text ends
```

6.1

- A. Loads in the string for Invalid Index
- B. Loads in endl addr, like in previous functions

6.2.

- A. Loads in the string for Index OOB
- B. Loads in endl addr, like in previous functions

6.3. Exit Block