Struct node:
- char data;
- int idata;
- struct node * nextPtr;
- struct node * prevPtr;

Special cases
- insert 0
- insert 2
- insert 4
- insert 7
- when *cPtr is middle
  - at element 1
  - at element 5

1. allocate new
2. check to see if valid
3. initialize

```c
struct node
{
    char data;
    int idata;
    struct node * nextPtr;
    struct node * prevPtr;
};
```
1. `newPtr -> prevPtr = *cPtr;`
2. `newPtr -> nextPtr = (*cPtr) -> nextPtr;`
3. `(*cPtr) -> nextPtr -> prevPtr = newPtr;`
   or
   `newPtr -> nextPtr -> prevPtr = newPtr`
4. `(*cPtr) -> nextPtr = newPtr;`

```
delete 3
```

```
delete before = go to correct place with seek. Now we are deleting
```

```
delete before
```

```
1. tempPtr = (*cPtr) -> prevPtr
2. (*cPtr) -> prevPtr = tempPtr -> prevPtr
3. tempPtr -> prevPtr -> nextPtr = *cPtr
4. free(tempPtr);
```