Salazar

Assignment 5

Assignment 5 (Auto-sorting list operations)

Completion requirements

Follow along with the in-class exercise on this, do your best to get it working, and turn in what you come up with here!

Be sure to include at least one test for each function or piece of functionality that should verify that your code is working! No slacking smile, you should start writing some tests before you write your implementations (just spend a few minutes thinking about the design and then write a few tests using natural language (English is preferred for me to be able to read it smile))

Create an array-based list or a linked-list (and a bonus for attempting both) that:

automatically inserts values in the correct position based on some order of sorting (perhaps ascending integers or lexicographical sorting of words)

efficiently searches for elements (likely binary search for the array list, but what about the linked-list?)

Make a chart to compare the algorithmic complexity (use Big-O notation) of your insert, remove, and search algorithms you used for your structures

Once you have implemented and tested your code, add to the README file what line(s) of code or inputs and outputs show your work meeting each of the above requirements (or better, include a small screen snip of where it meets the requirement!).

Note: This assignment is to get you to think about and practice with structures that have automatically enforced properties (perhaps auto sorted, since we know sorting can be expensive… and how these might be able to be spread out over time)

//Design

Will use an array-based list of pokemon

The arrange function will automatically rearrange the list alphabetically

Search function will iterate through the whole array till it finds what it is looking for.

Remove function, might use the search function , then delete the results

// test for functions

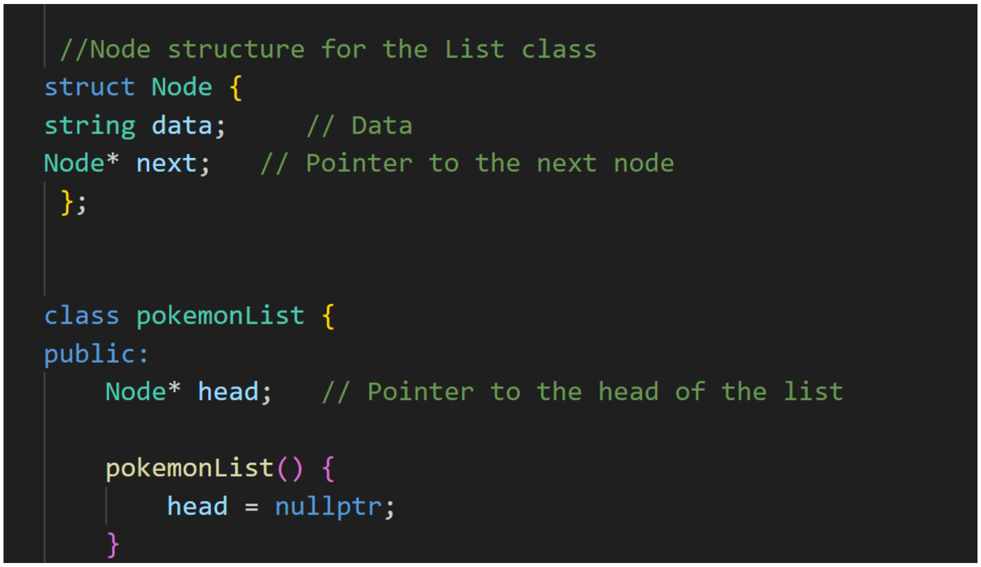
For the auto sorting, I'll just print out the list; it should now be alphabetized.

For the search function, I'll print out its position location, and compare to where it should be

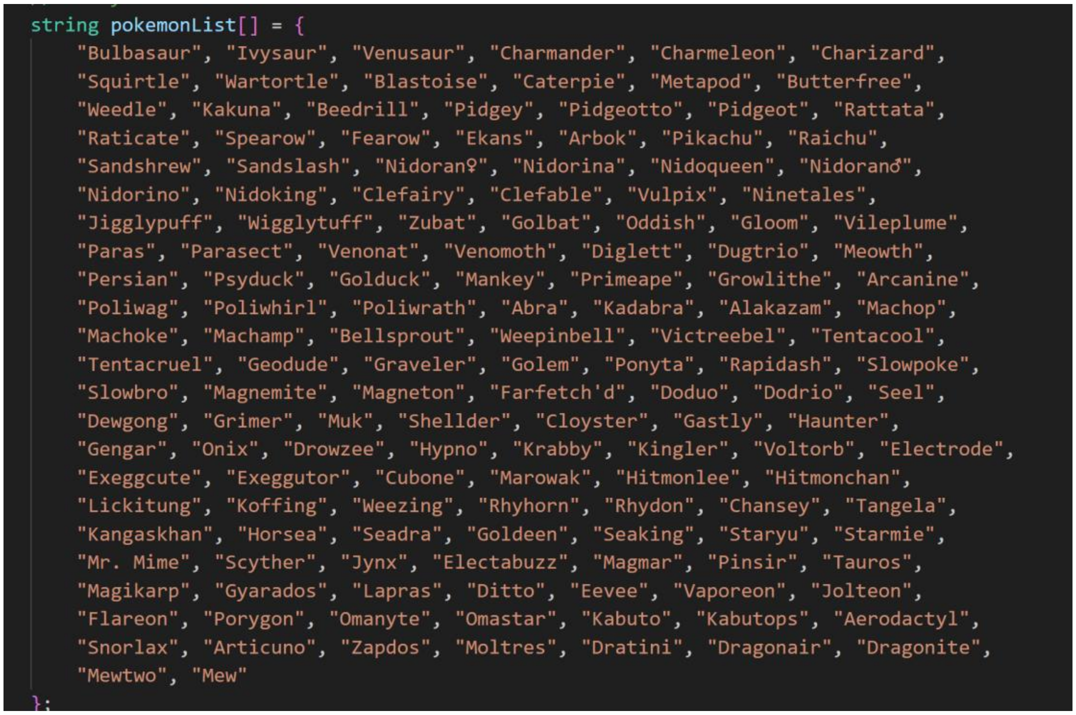
Remove function, after removing the pokemon I can check the lenthg of the list or print out the array and confirm its gone.

//requirements

//Array based list

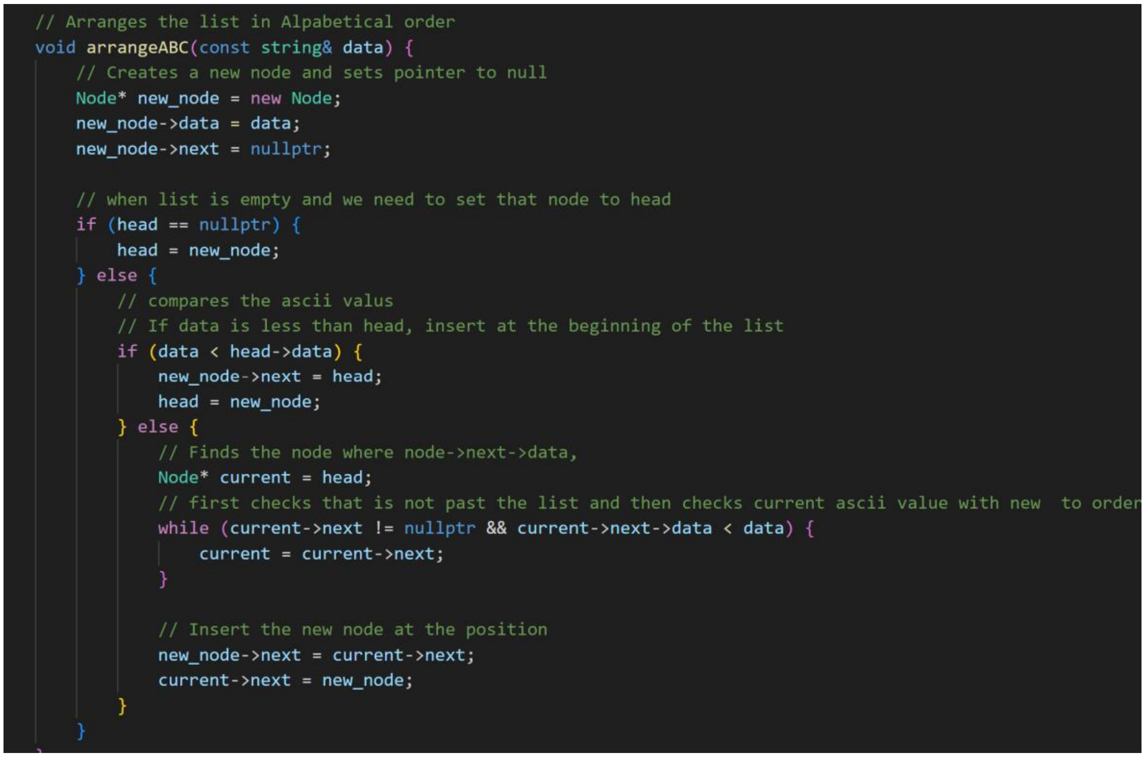


```
 //Node structure for the List class
struct Node {
string data;      // Data
Node* next;    // Pointer to the next node
 };


class pokemonList {
public:
    Node* head;    // Pointer to the head of the list

    pokemonList() {
        head = nullptr;
    }
```



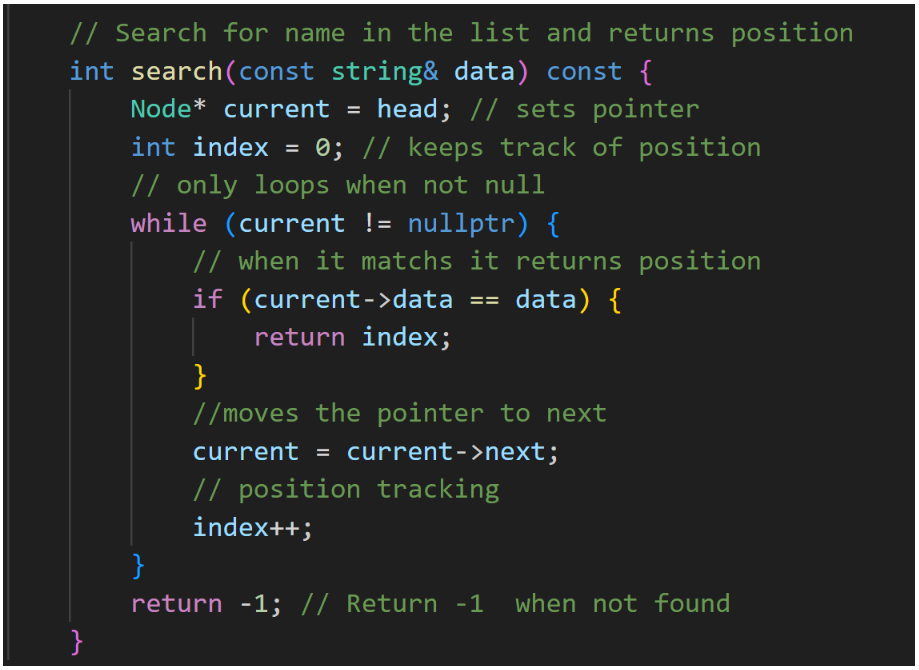
```
string pokemonList[] = {
    "Bulbasaur", "Ivysaur", "Venusaur", "Charmander", "Charmeleon", "Charizard",
    "Squirtle", "Wartortle", "Blastoise", "Caterpie", "Metapod", "Butterfree",
    "Weedle", "Kakuna", "Beedrill", "Pidgey", "Pidgeotto", "Pidgeot", "Rattata",
    "Raticate", "Spearow", "Fearow", "Ekans", "Arbok", "Pikachu", "Raichu",
    "Sandshrew", "Sandslash", "Nidoran♀", "Nidorina", "Nidoqueen", "Nidoran♂",
    "Nidorino", "Nidoking", "Clefairy", "Clefable", "Vulpix", "Ninetales",
    "Jigglypuff", "Wigglytuff", "Zubat", "Golbat", "Oddish", "Gloom", "Vileplume",
    "Paras", "Parasect", "Venonat", "Venomoth", "Diglett", "Dugtrio", "Meowth",
    "Persian", "Psyduck", "Golduck", "Mankey", "Primeape", "Growlithe", "Arcanine",
    "Poliwag", "Poliwhirl", "Poliwrath", "Abra", "Kadabra", "Alakazam", "Machop",
    "Machoke", "Machamp", "Bellsprout", "Weepinbell", "Victreebel", "Tentacool",
    "Tentacruel", "Geodude", "Graveler", "Golem", "Ponyta", "Rapidash", "Slowpoke",
    "Slowbro", "Magnemite", "Magneton", "Farfetch'd", "Doduo", "Dodrio", "Seel",
    "Dewgong", "Grimer", "Muk", "Shellder", "Cloyster", "Gastly", "Haunter",
    "Gengar", "Onix", "Drowzee", "Hypno", "Krabby", "Kingler", "Voltorb", "Electrode",
    "Exeggcute", "Exeggutor", "Cubone", "Marowak", "Hitmonlee", "Hitmonchan",
    "Lickitung", "Koffing", "Weezing", "Rhyhorn", "Rhydon", "Chansey", "Tangela",
    "Kangaskhan", "Horsea", "Seadra", "Goldeen", "Seaking", "Staryu", "Starmie",
    "Mr. Mime", "Scyther", "Jynx", "Electabuzz", "Magmar", "Pinsir", "Tauros",
    "Magikarp", "Gyarados", "Lapras", "Ditto", "Eevee", "Vaporeon", "Jolteon",
    "Flareon", "Porygon", "Omanyte", "Omastar", "Kabuto", "Kabutops", "Aerodactyl",
    "Snorlax", "Articuno", "Zapdos", "Moltres", "Dratini", "Dragonair", "Dragonite",
    "Mewtwo", "Mew"
};
```

// auto sort



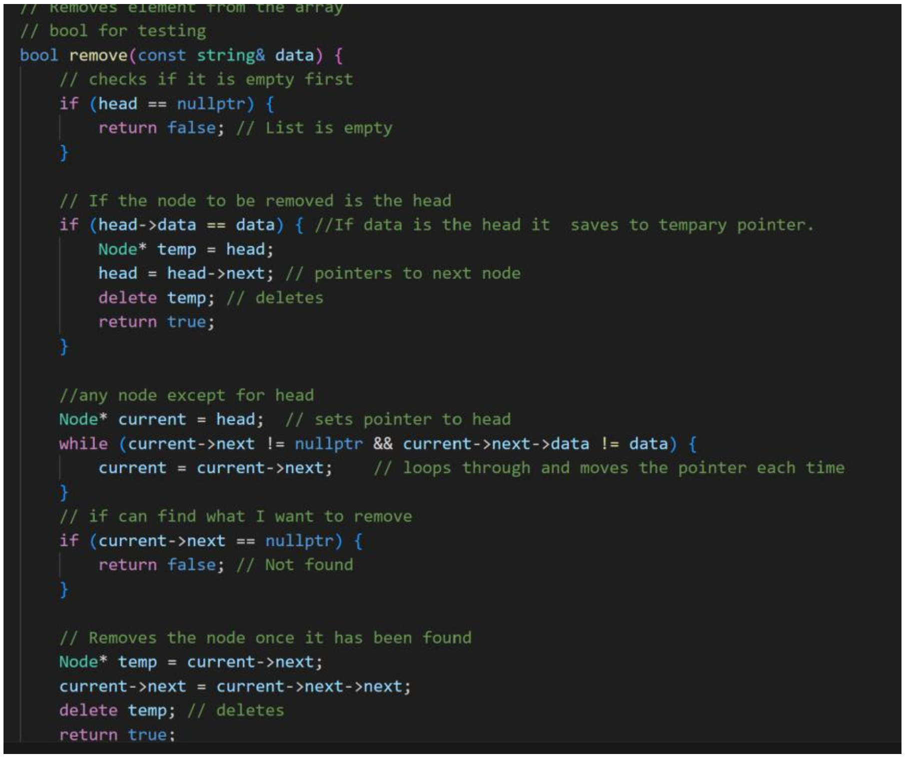
```
// Arranges the list in Alpabetical order
void arrangeABC(const string& data) {
    // Creates a new node and sets pointer to null
    Node* new_node = new Node;
    new_node->data = data;
    new_node->next = nullptr;

    // when list is empty and we need to set that node to head
    if (head == nullptr) {
        head = new_node;
    } else {
        // compares the ascii valus
        // If data is less than head, insert at the beginning of the list
        if (data < head->data) {
            new_node->next = head;
            head = new_node;
        } else {
            // Finds the node where node->next->data,
            Node* current = head;
            // first checks that is not past the list and then checks current ascii value with new  to order
            while (current->next != nullptr && current->next->data < data) {
                current = current->next;
            }

            // Insert the new node at the position
            new_node->next = current->next;
            current->next = new_node;
        }
    }
```

//search function



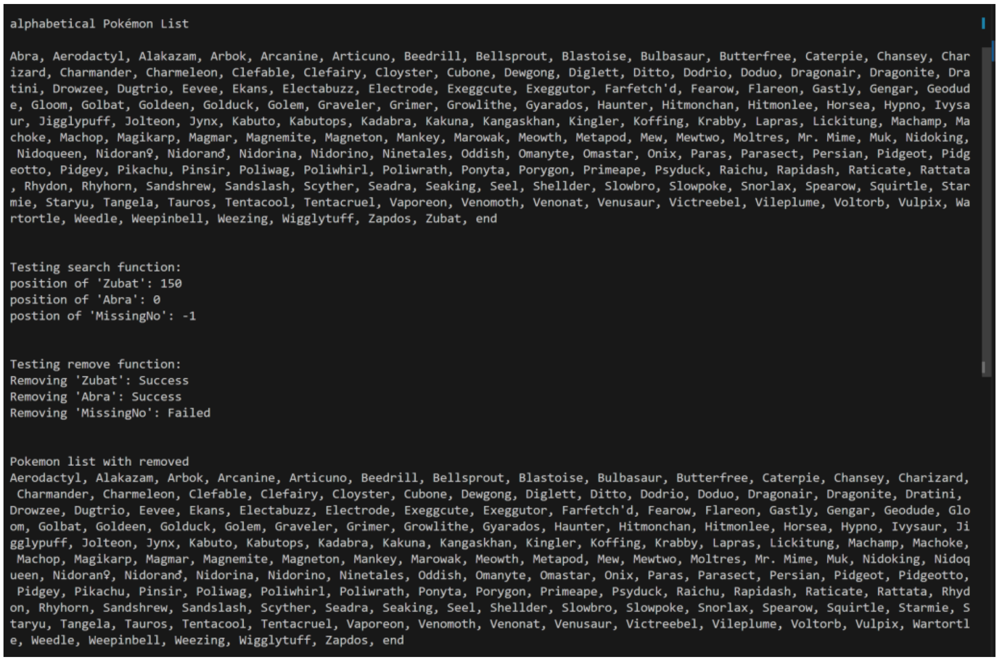
```
// Search for name in the list and returns position
int search(const string& data) const {
    Node* current = head; // sets pointer
    int index = 0; // keeps track of position
    // only loops when not null
    while (current != nullptr) {
        // when it matchs it returns position
        if (current->data == data) {
            return index;
        }
        //moves the pointer to next
        current = current->next;
        // position tracking
        index++;
    }
    return -1; // Return -1  when not found
}
```

// remove function



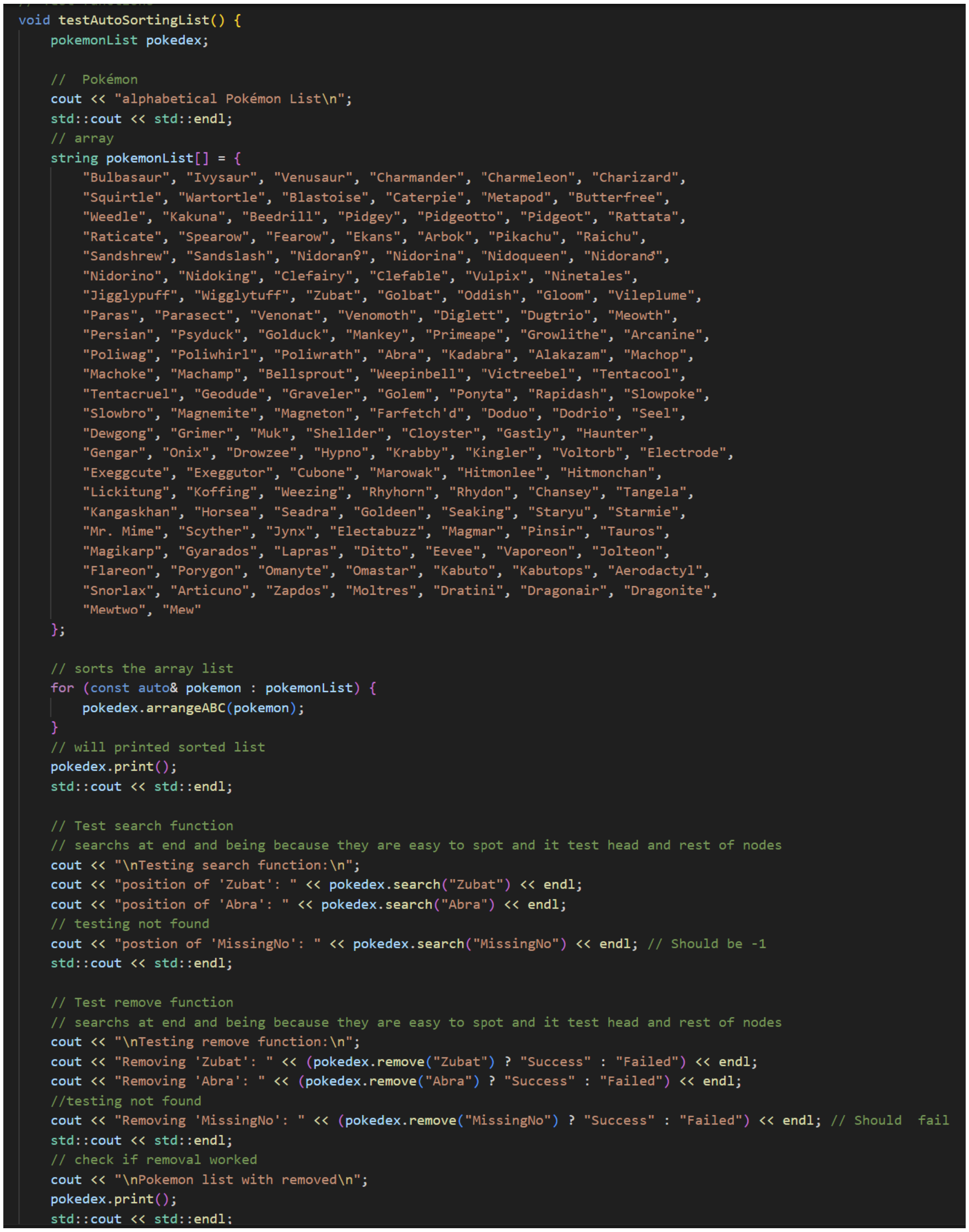
```
// Removes element from the array
// bool for testing
bool remove(const string& data) {
    // checks if it is empty first
    if (head == nullptr) {
        return false; // List is empty
    }

    // If the node to be removed is the head
    if (head->data == data) { //If data is the head it  saves to tempary pointer.
        Node* temp = head;
        head = head->next; // pointers to next node
        delete temp; // deletes
        return true;
    }

    //any node except for head
    Node* current = head;  // sets pointer to head
    while (current->next != nullptr && current->next->data != data) {
        current = current->next;    // loops through and moves the pointer each time
    }
    // if can find what I want to remove
    if (current->next == nullptr) {
        return false; // Not found
    }

    // Removes the node once it has been found
    Node* temp = current->next;
    current->next = current->next->next;
    delete temp; // deletes
    return true;
```

// results from testing



```
alphabetical Pokémon List

Abra, Aerodactyl, Alakazam, Arbok, Arcanine, Articuno, Beedrill, Bellsprout, Blastoise, Bulbasaur, Butterfree, Caterpie, Chansey, Char
izard, Charmander, Charmeleon, Clefable, Clefairy, Cloyster, Cubone, Dewgong, Diglett, Ditto, Dodrio, Doduo, Dragonair, Dragonite, Dra
tini, Drowzee, Dugtrio, Eevee, Ekans, Electabuzz, Electrode, Exeggcute, Exeggutor, Farfetch'd, Fearow, Flareon, Gastly, Gengar, Geodud
e, Gloom, Golbat, Goldeen, Golduck, Golem, Graveler, Grimer, Growlithe, Gyarados, Haunter, Hitmonchan, Hitmonlee, Horsea, Hypno, Ivysa
ur, Jigglypuff, Jolteon, Jynx, Kabuto, Kabutops, Kadabra, Kakuna, Kangaskhan, Kingler, Koffing, Krabby, Lapras, Lickitung, Machamp, Ma
choke, Machop, Magikarp, Magmar, Magnemite, Magneton, Mankey, Marowak, Meowth, Metapod, Mew, Mewtwo, Moltres, Mr. Mime, Muk, Nidoking,
 Nidoqueen, Nidoran♀, Nidoran♂, Nidorina, Nidorino, Ninetales, Oddish, Omanyte, Omastar, Onix, Paras, Parasect, Persian, Pidgeot, Pidg
eotto, Pidgey, Pikachu, Pinsir, Poliwag, Poliwhirl, Poliwrath, Ponyta, Porygon, Primeape, Psyduck, Raichu, Rapidash, Raticate, Rattata
, Rhydon, Rhyhorn, Sandshrew, Sandslash, Scyther, Seadra, Seaking, Seel, Shellder, Slowbro, Slowpoke, Snorlax, Spearow, Squirtle, Star
mie, Staryu, Tangela, Tauros, Tentacool, Tentacruel, Vaporeon, Venomoth, Venonat, Venusaur, Victreebel, Vileplume, Voltorb, Vulpix, Wa
rtortle, Weedle, Weepinbell, Weezing, Wigglytuff, Zapdos, Zubat, end


Testing search function:
position of 'Zubat': 150
position of 'Abra': 0
postion of 'MissingNo': -1


Testing remove function:
Removing 'Zubat': Success
Removing 'Abra': Success
Removing 'MissingNo': Failed


Pokemon list with removed
Aerodactyl, Alakazam, Arbok, Arcanine, Articuno, Beedrill, Bellsprout, Blastoise, Bulbasaur, Butterfree, Caterpie, Chansey, Charizard,
 Charmander, Charmeleon, Clefable, Clefairy, Cloyster, Cubone, Dewgong, Diglett, Ditto, Dodrio, Doduo, Dragonair, Dragonite, Dratini,
Drowzee, Dugtrio, Eevee, Ekans, Electabuzz, Electrode, Exeggcute, Exeggutor, Farfetch'd, Fearow, Flareon, Gastly, Gengar, Geodude, Glo
om, Golbat, Goldeen, Golduck, Golem, Graveler, Grimer, Growlithe, Gyarados, Haunter, Hitmonchan, Hitmonlee, Horsea, Hypno, Ivysaur, Ji
gglypuff, Jolteon, Jynx, Kabuto, Kabutops, Kadabra, Kakuna, Kangaskhan, Kingler, Koffing, Krabby, Lapras, Lickitung, Machamp, Machoke,
 Machop, Magikarp, Magmar, Magnemite, Magneton, Mankey, Marowak, Meowth, Metapod, Mew, Mewtwo, Moltres, Mr. Mime, Muk, Nidoking, Nidoq
ueen, Nidoran♀, Nidoran♂, Nidorina, Nidorino, Ninetales, Oddish, Omanyte, Omastar, Onix, Paras, Parasect, Persian, Pidgeot, Pidgeotto,
 Pidgey, Pikachu, Pinsir, Poliwag, Poliwhirl, Poliwrath, Ponyta, Porygon, Primeape, Psyduck, Raichu, Rapidash, Raticate, Rattata, Rhyd
on, Rhyhorn, Sandshrew, Sandslash, Scyther, Seadra, Seaking, Seel, Shellder, Slowbro, Slowpoke, Snorlax, Spearow, Squirtle, Starmie, S
taryu, Tangela, Tauros, Tentacool, Tentacruel, Vaporeon, Venomoth, Venonat, Venusaur, Victreebel, Vileplume, Voltorb, Vulpix, Wartortl
e, Weedle, Weepinbell, Weezing, Wigglytuff, Zapdos, end
```

//testing is all done in one big function that test each of the functions to save space from list



```cpp
void testAutoSortingList() {
    pokemonList pokedex;

    //  Pokémon
    cout << "alphabetical Pokémon List\n";
    std::cout << std::endl;
    // array
    string pokemonList[] = {
        "Bulbasaur", "Ivysaur", "Venusaur", "Charmander", "Charmeleon", "Charizard",
        "Squirtle", "Wartortle", "Blastoise", "Caterpie", "Metapod", "Butterfree",
        "Weedle", "Kakuna", "Beedrill", "Pidgey", "Pidgeotto", "Pidgeot", "Rattata",
        "Raticate", "Spearow", "Fearow", "Ekans", "Arbok", "Pikachu", "Raichu",
        "Sandshrew", "Sandslash", "Nidoran♀", "Nidorina", "Nidoqueen", "Nidoran♂",
        "Nidorino", "Nidoking", "Clefairy", "Clefable", "Vulpix", "Ninetales",
        "Jigglypuff", "Wigglytuff", "Zubat", "Golbat", "Oddish", "Gloom", "Vileplume",
        "Paras", "Parasect", "Venonat", "Venomoth", "Diglett", "Dugtrio", "Meowth",
        "Persian", "Psyduck", "Golduck", "Mankey", "Primeape", "Growlithe", "Arcanine",
        "Poliwag", "Poliwhirl", "Poliwrath", "Abra", "Kadabra", "Alakazam", "Machop",
        "Machoke", "Machamp", "Bellsprout", "Weepinbell", "Victreebel", "Tentacool",
        "Tentacruel", "Geodude", "Graveler", "Golem", "Ponyta", "Rapidash", "Slowpoke",
        "Slowbro", "Magnemite", "Magneton", "Farfetch'd", "Doduo", "Dodrio", "Seel",
        "Dewgong", "Grimer", "Muk", "Shellder", "Cloyster", "Gastly", "Haunter",
        "Gengar", "Onix", "Drowzee", "Hypno", "Krabby", "Kingler", "Voltorb", "Electrode",
        "Exeggcute", "Exeggutor", "Cubone", "Marowak", "Hitmonlee", "Hitmonchan",
        "Lickitung", "Koffing", "Weezing", "Rhyhorn", "Rhydon", "Chansey", "Tangela",
        "Kangaskhan", "Horsea", "Seadra", "Goldeen", "Seaking", "Staryu", "Starmie",
        "Mr. Mime", "Scyther", "Jynx", "Electabuzz", "Magmar", "Pinsir", "Tauros",
        "Magikarp", "Gyarados", "Lapras", "Ditto", "Eevee", "Vaporeon", "Jolteon",
        "Flareon", "Porygon", "Omanyte", "Omastar", "Kabuto", "Kabutops", "Aerodactyl",
        "Snorlax", "Articuno", "Zapdos", "Moltres", "Dratini", "Dragonair", "Dragonite",
        "Mewtwo", "Mew"
    };

    // sorts the array list
    for (const auto& pokemon : pokemonList) {
        pokedex.arrangeABC(pokemon);
    }
    // will printed sorted list
    pokedex.print();
    std::cout << std::endl;

    // Test search function
    // searchs at end and being because they are easy to spot and it test head and rest of nodes
    cout << "\nTesting search function:\n";
    cout << "position of 'Zubat': " << pokedex.search("Zubat") << endl;
    cout << "position of 'Abra': " << pokedex.search("Abra") << endl;
    // testing not found
    cout << "postion of 'MissingNo': " << pokedex.search("MissingNo") << endl; // Should be -1
    std::cout << std::endl;

    // Test remove function
    // searchs at end and being because they are easy to spot and it test head and rest of nodes
    cout << "\nTesting remove function:\n";
    cout << "Removing 'Zubat': " << (pokedex.remove("Zubat") ? "Success" : "Failed") << endl;
    cout << "Removing 'Abra': " << (pokedex.remove("Abra") ? "Success" : "Failed") << endl;
    //testing not found
    cout << "Removing 'MissingNo': " << (pokedex.remove("MissingNo") ? "Success" : "Failed") << endl; // Should  fail
    std::cout << std::endl;
    // check if removal worked
    cout << "\nPokemon list with removed\n";
    pokedex.print();
    std::cout << std::endl;
```

// in terms of complexity all the functions are going to be the same on O(n).

The sort function needs to go through the whole list.

If the Pokemon is not in the list or at the very end, the search function must go through the whole list.

The remove function also does the same while looking through.