Alex Simak Projects

High school

- Java (Highschool Project): Hangman
  - File I/O, User Input, String manipulation
- C (APL): Reverse Polish Notation Calculator
  - Basic arithmetic operations, graphing and a variety of formulas

Semester 1

- Python: Music Library
  - Pull data from DB, store based on genre, year, artist and create playlists based on user input
- Python: Conway's Game of Life
  - Two-dimensional lists, mutability, and creating and calling functions
- Python: Maze Solver
  - File I/O, two-dimensional lists, creating and calling functions, and recursion.

Semester 2

- C++: Urban Heat Islands
  - Calling Multiple functions, 2D arrays, File I/O
- C++: Dragon Wars
  - Classes and instantiation of classes using constructors, vectors, File I/O, makefile
- C++: UMBC Trains
  - Classes/ instantiation of constructors, linked list, I/O, Overloaded operators and enums
- C++: UMBC Shipping
  - Inheritance and Polymorphism, File I/O, Driver Class
- C++: Star Wars
  - Templated data structure (queue or stack), overloaded operators

Semester 3

- C++: Circular Buffer of Circular Buffers
  - object-oriented design, dynamic memory allocation (of arrays) and pointer manipulation
- C++: Sally Forth
  - Build a compiler that starts off working with arithmetic operations, then expanding it to handle stack operations, variables, comparison and logical operators, if statements and loops
- C++: Quadtrees for Life
  - Give a basic understanding of a tree-based data structure
- C++: Median Heaps
  - Binary Heaps, Templates and function pointers
- C++: Incremental Rehash
  - Implementation of a Hash Table

Semester 4

- Machine Code: Bomb Lab
  - Principles of Machine-level programs, general debugging and reverse engineering
- SQL/Python: Linking
  - Links a database schema, to a query language, (SQL) using python.

Semester 6

- Artificial Intelligence: Dijkstra and A* (Python)
  - Implemented Dijkstra's and A* to build a pathfinding algorithm for a terrain mapped as a bitmap.
- Artificial Intelligence: Connect 4
  - Implemented a game playing agent that plays connect four with the Minimax algorithm allowing the player to choose the level of difficulty to play the game.
- Artificial Intelligence: Random Forest Classification
  - Used the Random Forest Classifier in the scikitlearn package in Python in order to predict whether or not someone makes more than or less than $50k.
- C: Chess and Tic Tac Toe (Operating Systems)
  - Implemented a kernel module that can be used to play against the user in a game of tic-tac-toe. This module will implement a virtual character device (with a corresponding entry in /dev that the user can open a file descriptor to), which will be interacted with by the user using normal file read and write operations.

Side Projects

- Python: I was searching for a new car, because I was commuting to school and work and I would constantly check the website Cars.com for any new cars that I preferred. Instead of constantly checking, I created a web scraper that automatically scrapes the website using the BeautifulSoup Library and if a new car, with my desires filters appears, it will send me an email with the new car/s.
- Python: I created a simple game using a GUI for the user interface using pygame, which uses the keyboard as the interaction feature, which the player uses the arrows to dodge obstacles and tries to last longer, the game gets faster as the time progresses which makes it harder.
- MERN: MongoDB, Express, React, Node
  - Created a simple website using HTML/CSS and React. The website took in user input (login information) and was able to link the information with the database which either allowed or denied user permission to the application.
  - https://blog.hyperiondev.com/index.php/2018/09/10/everything-need-know-mern-stack/ (for me)
- JavaScript/Unity: The Impossible Game
  - I developed a mobile game for the android phone which resembled the game Mario. The player would use the touch screen components on their device to move the player through a 2D world to avoid the obstacles coming at them. The objective of the game is to get passed all the obstacles and reach the finish line.
- React/AWS:
  - Created a very basic login page using react that is linked with AWS to publish and maintain the website.
- jQuery:
  - Assisted my father to manipulate his website, using jQuery to automatically adjust the CSS animation and event handling to make it mobile application friendly.
- Machine Learning Experience:

- Very basic model which inputs stock data using pandas data reader, takes over 10 years of data (depending the length of time the stock has been around) and based on that builds and trains a model based on the close price for that specific stock, then represented via a graph showing what the actual data shows vs the expected based on the trained model.

This is just a little project i worked on for a couple weeks and it actually helped me which is why i value this project. I was searching for a new car, because I was commuting to school and work and I would constantly check the website Cars.com for any new cars that I preferred. Instead of constantly checking, I created a web scraper that automatically scrapes the website using the BeautifulSoup Library and if a new car, with my desires filters appears, it will update a csv file and then send me an email with the new car/s. One day I was sitting at a stop light in my old 1997 Corolla when I got an email with a new car and that is the same day I bought it, and ever since I have been thankful that I got that notification and was able to buy the car before anyone else could.

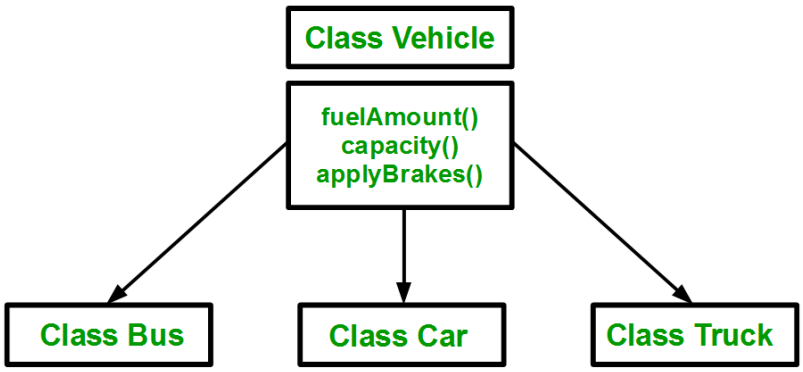
I also worked with a friend to try and develop a website where people would sign up to get online help with any technical difficulties they had. It would be kind of like fiver but it would be strictly with tech issues and would target older people so instead of nagging their children/ grandchildren people on the internet could help them out. We got to the portion of making a log in, linking it with a database and then uploading it with AWS but then the school semester started and we had to put the project on the backburner. We used MERN and AWS to manage and maintain the website.

DATA STRUCTURES:

1. **Linked list vs array:**
   - Array: Fixed Length
   - Linked List: Dynamic size and ease of insertion/deletion
   - Accessing an element in an array is fast, while Linked list takes linear time, so it is quite a bit slower

2. **Enum:** used to assign names to integral constants

3. **Inheritance:**
   a. We have a class vehicle which can hold instances such as color, mpg, wheels etc, and within the vehicle there are subclasses such as a bus/car/truck.
   Pro: Avoids reusing the same code. Only need to change the super class, in this case vehicle and the subclasses will inherit those properties.



4. **Polymorphism:** Compile time vs runtime
    a. Compile: Function overloading and operator overloading
    b. Runtime: function overriding
5. **Queue:**
    a. Push()/pop(), size, front(),back()
6. **Circular Queue/buffer**: its basically a fixed size circle, end to end that holds data.
    a. Benefits: don't need to shuffle info around when things are removed, O(1) – constant
7. **Binary Heap:**
    a. Search: O(n)
    b. Insert: O(1)
    c. (All levels are filled except possibly the last level and the last level has all keys as left as possible). This property of Binary Heap makes them suitable to be stored in an array.
    d. A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at root must be minimum among all keys present in Binary Heap. The same property must be recursively true for all nodes in Binary Tree. Max Binary Heap is similar to MinHeap.
8. **Hash Table:**
    a. Maps keys to values
    b. Uses a hash function to make an index into an array (buckets slots)
    c. Cons: hash collisions depending on how strong the hash function is

---

Coding examples:

1. Write a program in any language that will continuously print prime numbers until user presses CTRL C.

```
x = 0
while(True):
```

```
  # all prime numbers are greater than 1
  if x > 1:
        # basically we want to see if we can mod the outer loop by any number below it to see if
it divides, if not it is a prime
     for i in range(2, x):
        if (x % i) == 0:
              break
     else:
        print(x)
  x += 1
```

2. Sum a string of integers.

```
str1 = "2a2ale4x123"
# A temporary str1ing
temp = ""

# holds sum of all numbers
# present in the str1ing
Sum = 0

# read each character in input string
for ch in str1:

   # if current character is a digit
   if (ch.isdigit()):
      temp += ch

   # if current character is an alphabet (elif)

   else:

      # increment Sum by number found earlier(if any)
      Sum += int(temp)
      # reset temporary str1ing to empty
      temp = "0"

# atoi(temp.c_str1()) takes care of trailing numbers
print(Sum + int(temp))
```

3. What is the purpose of the "volatile" keyword in C?
    a. **variable** when it is declared. It tells the compiler that the value of the **variable** may change at any time
4. Implement stack with two queues

```
using namespace std;
```

```
class Stack{
        #initialize the queues
        queue<int> q1, q2;
        int curr_size;
}
public:
        Stack(){
                curr_size = 0;
        }
        void push(int x){
                curr_size++;
                q2.push(x);

                        while(!q1.empty()){
                        q2.push(q1.front());
                        q1.pop();
                }
                queue<int> q = q1;
                q1=q2;
                q2 = q;
        }
        void pop(){
                if(q1.empty())
                        return;
                q1.pop();
                curr_size--;
        }
        int top(){
                if(q1.empty())
                        return -1;
                return q1.front();
        }
        int size(){
                return curr_size;
        }

// main driver
        int main() {
   Stack s;
   s.push(1);
   s.push(2);
   s.push(3);

   cout << "current size: " << s.size()
        << endl;
   cout << s.top() << endl;
   s.pop();
   cout << s.top() << endl;
```

```
    s.pop();
    cout << s.top() << endl;

    cout << "current size: " << s.size()
         << endl;
    return 0;
}
```

5. Inode: data structure on a filesystem on Linux and other Unix-like operating systems that stores all the information about a file except its name and its actual data
6. Fib Sequence:

```
#0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

def Fibonacci(n):
    # First Fibonacci number is 0
    if n==1:
        return 0
    # Second Fibonacci number is 1
    elif n==2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)

# Driver Program
print(Fibonacci(9))
```

7. Factorial:

```
n = 5
fact = 1

for i in range(1,n+1):
    fact = fact * i

print (fact)
```

8. Reverse String

```
s = "Alex"
str = ""
for i in s:
  str = i + str
print(str)
```

9. Linux Commands
    a. Ls – lists files
    b. Df- amount of free disk space
    c. Ed – text editor
    d. Cp – copy
    e. Mv – move
    f. Rm – remove
10. If characters are unique in a string

```
 # If at any time we encounter 2
   # same characters, return false
   for i in range(len(str)):
      for j in range(i + 1,len(str)):
         if(str[i] == str[j]):
            return False;

   # If no duplicate characters
   # encountered, return true
   return True;
```