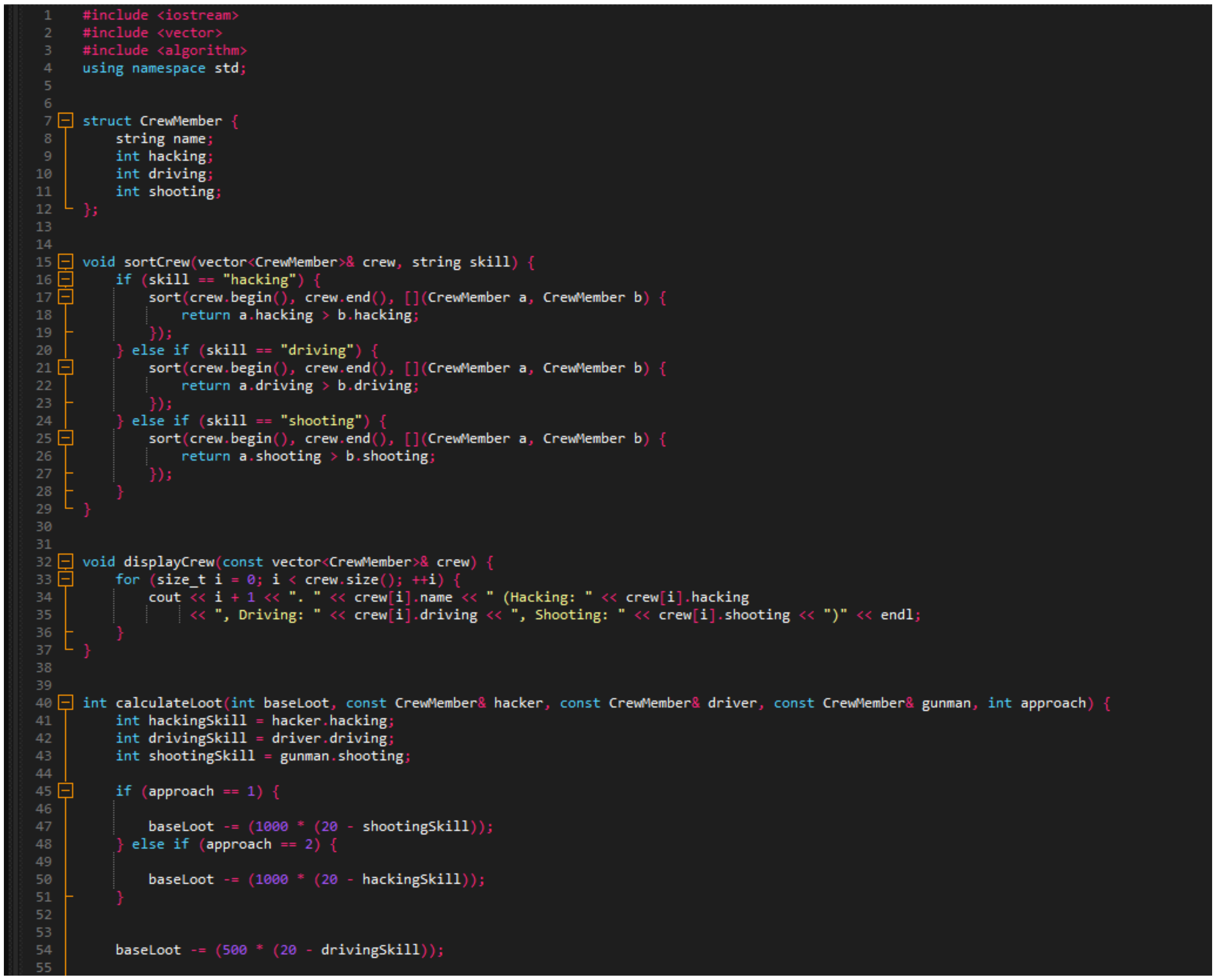
Villas, Kaith Angel D. 15/12/2024
CPE21S2 Sir Dennis Nava

Write a C++ or Python program of a Text-based game. The Program should implement a data structure, sorting algorithm,searching algorithm and a specific algorithm of your choice (like Greedy algorithm, Divide and Conquer or Dynamic programming).
Please provide the discussion concerning why did you choose the algorithms implemented in your program.

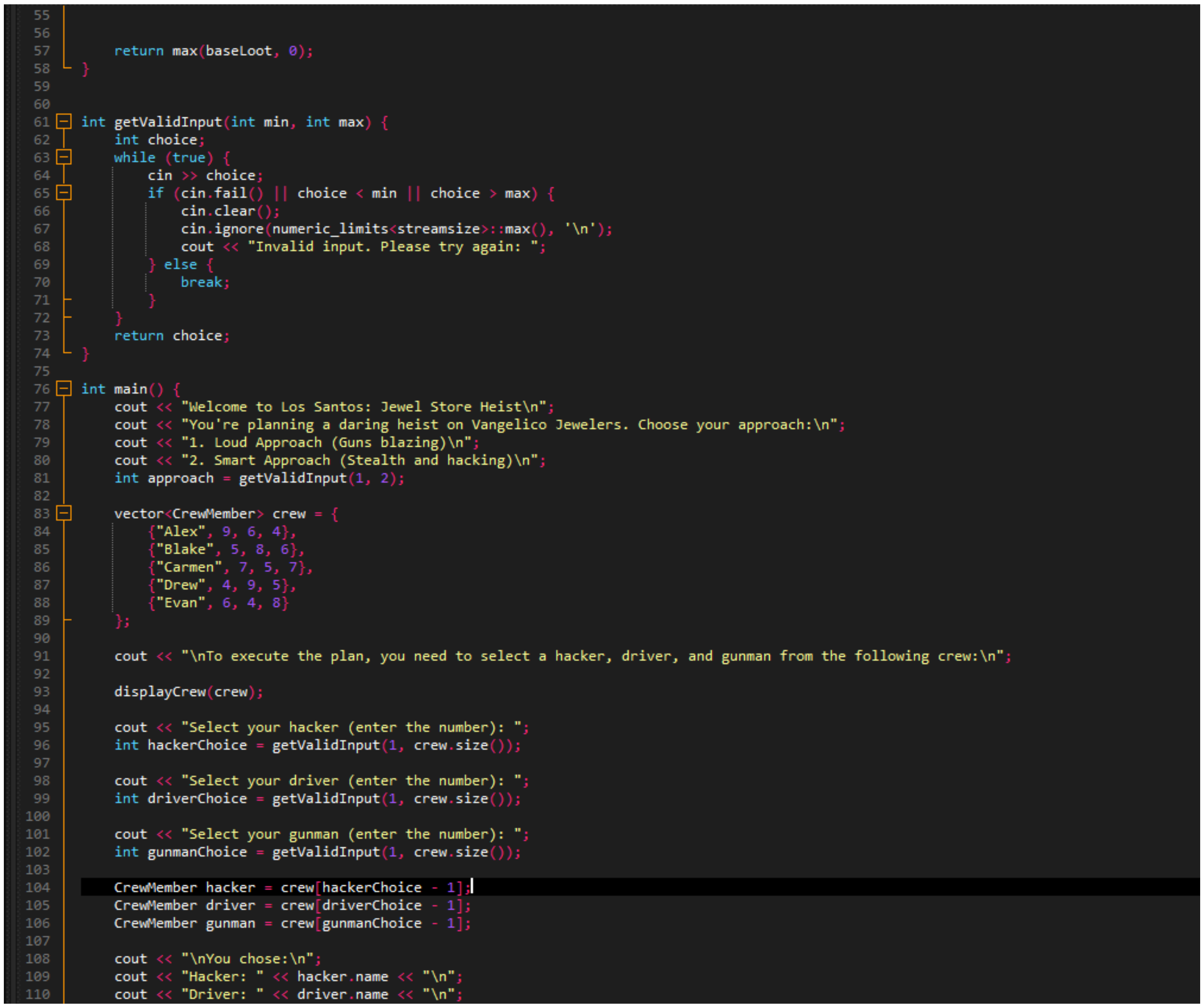
Please provide the screenshots of the source code and output. Please include your name and section in the documentation.
Please also provide the link where source codes can be downloaded. (Like Google drive or onlinegdb)

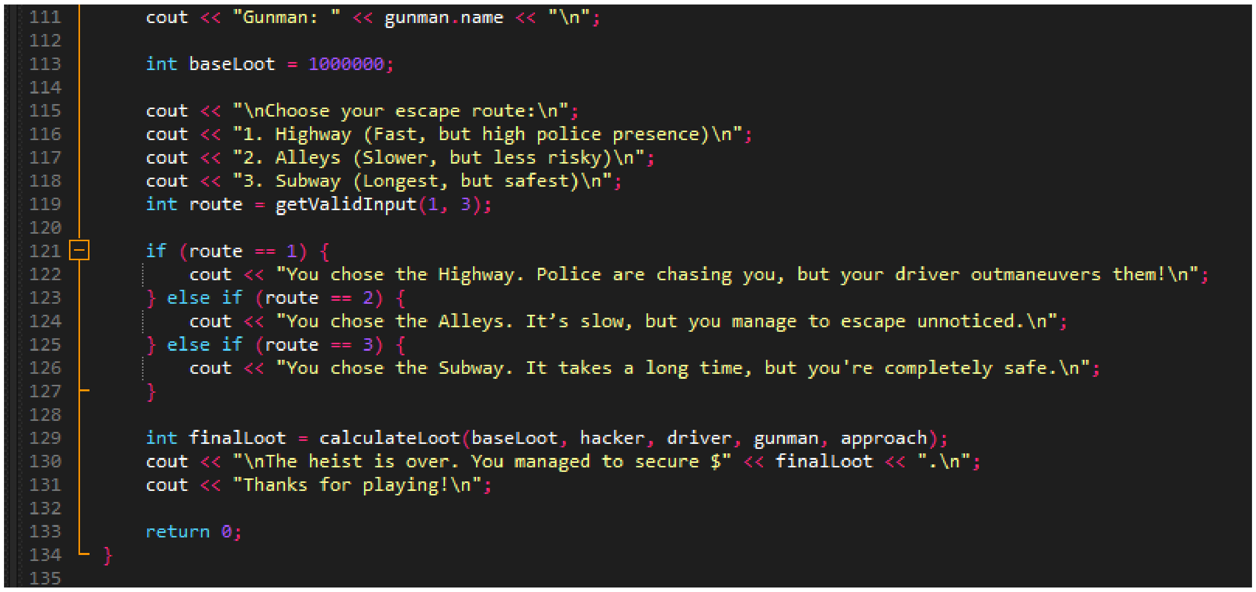
CODE AND OUTPUT

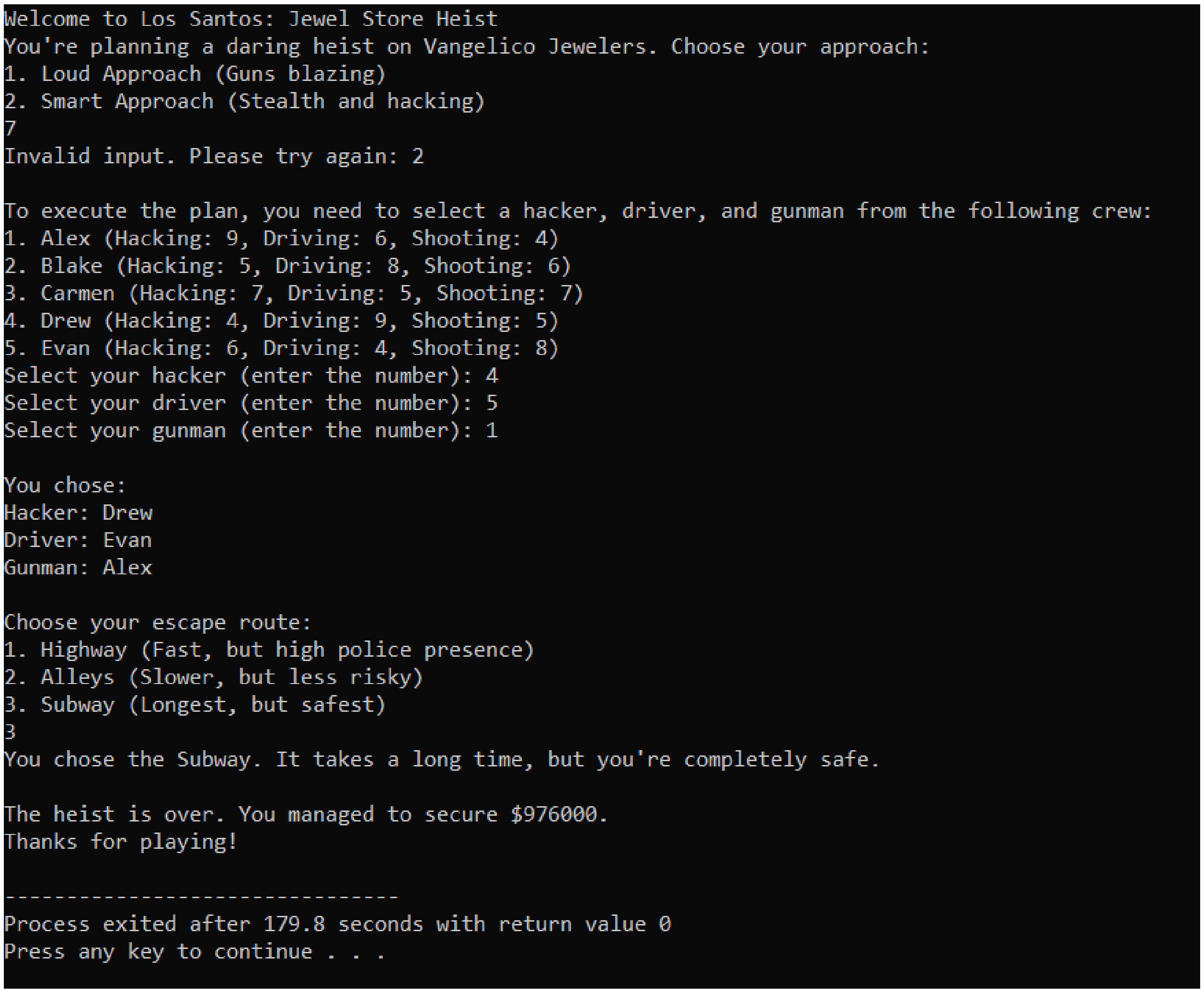
https://onlinegdb.com/y29EboIQG



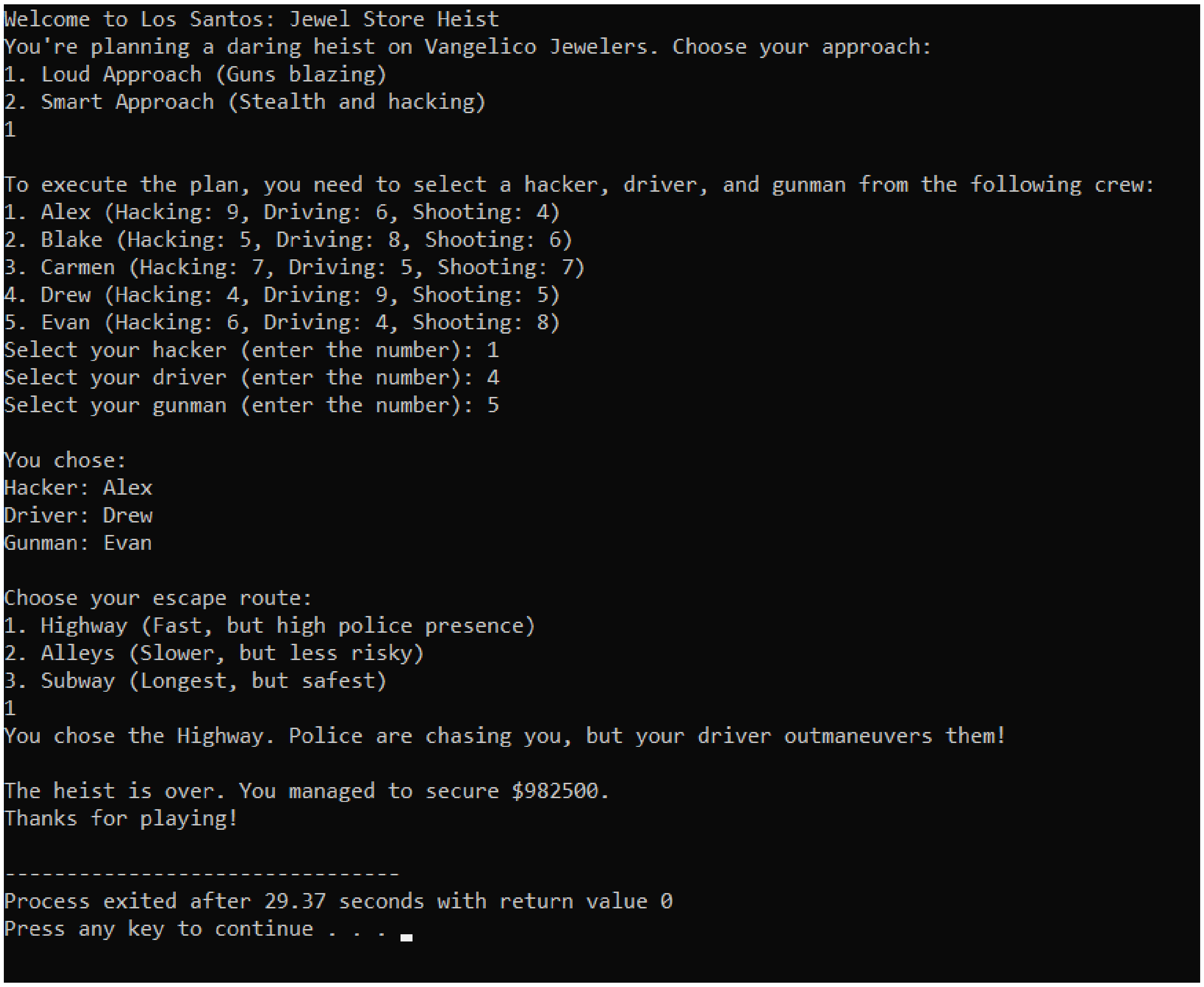
```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;


struct CrewMember {
    string name;
    int hacking;
    int driving;
    int shooting;
};


void sortCrew(vector<CrewMember>& crew, string skill) {
    if (skill == "hacking") {
        sort(crew.begin(), crew.end(), [](CrewMember a, CrewMember b) {
            return a.hacking > b.hacking;
        });
    } else if (skill == "driving") {
        sort(crew.begin(), crew.end(), [](CrewMember a, CrewMember b) {
            return a.driving > b.driving;
        });
    } else if (skill == "shooting") {
        sort(crew.begin(), crew.end(), [](CrewMember a, CrewMember b) {
            return a.shooting > b.shooting;
        });
    }
}


void displayCrew(const vector<CrewMember>& crew) {
    for (size_t i = 0; i < crew.size(); ++i) {
        cout << i + 1 << ". " << crew[i].name << " (Hacking: " << crew[i].hacking
            << ", Driving: " << crew[i].driving << ", Shooting: " << crew[i].shooting << ")" << endl;
    }
}


int calculateLoot(int baseLoot, const CrewMember& hacker, const CrewMember& driver, const CrewMember& gunman, int approach) {
    int hackingSkill = hacker.hacking;
    int drivingSkill = driver.driving;
    int shootingSkill = gunman.shooting;

    if (approach == 1) {

        baseLoot -= (1000 * (20 - shootingSkill));
    } else if (approach == 2) {

        baseLoot -= (1000 * (20 - hackingSkill));
    }


    baseLoot -= (500 * (20 - drivingSkill));

```


```cpp


    return max(baseLoot, 0);
}


int getValidInput(int min, int max) {
    int choice;
    while (true) {
        cin >> choice;
        if (cin.fail() || choice < min || choice > max) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid input. Please try again: ";
        } else {
            break;
        }
    }
    return choice;
}

int main() {
    cout << "Welcome to Los Santos: Jewel Store Heist\n";
    cout << "You're planning a daring heist on Vangelico Jewelers. Choose your approach:\n";
    cout << "1. Loud Approach (Guns blazing)\n";
    cout << "2. Smart Approach (Stealth and hacking)\n";
    int approach = getValidInput(1, 2);

    vector<CrewMember> crew = {
        {"Alex", 9, 6, 4},
        {"Blake", 5, 8, 6},
        {"Carmen", 7, 5, 7},
        {"Drew", 4, 9, 5},
        {"Evan", 6, 4, 8}
    };

    cout << "\nTo execute the plan, you need to select a hacker, driver, and gunman from the following crew:\n";

    displayCrew(crew);

    cout << "Select your hacker (enter the number): ";
    int hackerChoice = getValidInput(1, crew.size());

    cout << "Select your driver (enter the number): ";
    int driverChoice = getValidInput(1, crew.size());

    cout << "Select your gunman (enter the number): ";
    int gunmanChoice = getValidInput(1, crew.size());

    CrewMember hacker = crew[hackerChoice - 1];
    CrewMember driver = crew[driverChoice - 1];
    CrewMember gunman = crew[gunmanChoice - 1];

    cout << "\nYou chose:\n";
    cout << "Hacker: " << hacker.name << "\n";
    cout << "Driver: " << driver.name << "\n";
```


```cpp
    cout << "Gunman: " << gunman.name << "\n";

    int baseLoot = 1000000;

    cout << "\nChoose your escape route:\n";
    cout << "1. Highway (Fast, but high police presence)\n";
    cout << "2. Alleys (Slower, but less risky)\n";
    cout << "3. Subway (Longest, but safest)\n";
    int route = getValidInput(1, 3);

    if (route == 1) {
        cout << "You chose the Highway. Police are chasing you, but your driver outmaneuvers them!\n";
    } else if (route == 2) {
        cout << "You chose the Alleys. It’s slow, but you manage to escape unnoticed.\n";
    } else if (route == 3) {
        cout << "You chose the Subway. It takes a long time, but you're completely safe.\n";
    }

    int finalLoot = calculateLoot(baseLoot, hacker, driver, gunman, approach);
    cout << "\nThe heist is over. You managed to secure $" << finalLoot << ".\n";
    cout << "Thanks for playing!\n";

    return 0;
}

```



```
Welcome to Los Santos: Jewel Store Heist
You're planning a daring heist on Vangelico Jewelers. Choose your approach:
1. Loud Approach (Guns blazing)
2. Smart Approach (Stealth and hacking)
7
Invalid input. Please try again: 2

To execute the plan, you need to select a hacker, driver, and gunman from the following crew:
1. Alex (Hacking: 9, Driving: 6, Shooting: 4)
2. Blake (Hacking: 5, Driving: 8, Shooting: 6)
3. Carmen (Hacking: 7, Driving: 5, Shooting: 7)
4. Drew (Hacking: 4, Driving: 9, Shooting: 5)
5. Evan (Hacking: 6, Driving: 4, Shooting: 8)
Select your hacker (enter the number): 4
Select your driver (enter the number): 5
Select your gunman (enter the number): 1

You chose:
Hacker: Drew
Driver: Evan
Gunman: Alex

Choose your escape route:
1. Highway (Fast, but high police presence)
2. Alleys (Slower, but less risky)
3. Subway (Longest, but safest)
3
You chose the Subway. It takes a long time, but you're completely safe.

The heist is over. You managed to secure $976000.
Thanks for playing!

--------------------------------
Process exited after 179.8 seconds with return value 0
Press any key to continue . . .
```



```
Welcome to Los Santos: Jewel Store Heist
You're planning a daring heist on Vangelico Jewelers. Choose your approach:
1. Loud Approach (Guns blazing)
2. Smart Approach (Stealth and hacking)
1

To execute the plan, you need to select a hacker, driver, and gunman from the following crew:
1. Alex (Hacking: 9, Driving: 6, Shooting: 4)
2. Blake (Hacking: 5, Driving: 8, Shooting: 6)
3. Carmen (Hacking: 7, Driving: 5, Shooting: 7)
4. Drew (Hacking: 4, Driving: 9, Shooting: 5)
5. Evan (Hacking: 6, Driving: 4, Shooting: 8)
Select your hacker (enter the number): 1
Select your driver (enter the number): 4
Select your gunman (enter the number): 5

You chose:
Hacker: Alex
Driver: Drew
Gunman: Evan

Choose your escape route:
1. Highway (Fast, but high police presence)
2. Alleys (Slower, but less risky)
3. Subway (Longest, but safest)
1
You chose the Highway. Police are chasing you, but your driver outmaneuvers them!

The heist is over. You managed to secure $982500.
Thanks for playing!

--------------------------------
Process exited after 29.37 seconds with return value 0
Press any key to continue . . .
```

DISCUSSION

The algorithms used for the heist program are chosen to fit the demands of the project while having simple and easy-to-read code. The crew list is displayed to the players in a default order to let them choose between various crew members that have shown their hacking, driving, and shooting abilities. They can compare crew members by their strength and pick the ones with appropriate choices without the overhead functionality of sorting. This uses a combination of quicksort, heapsort and insertion sort. This approach ensures efficient sorting with good average-case performance across different types of input data, while protecting against the worst-case scenario of quicksort by switching to heapsort if needed.

Another very important component to make the program run as seamlessly as possible is input validation. The custom function getValidInput(min, max) is applied to ensure all inputs of crew selection and choice for an exit route are within a reasonable value range. It will loop around asking for the input over and over again until such time that a valid option has been provided.

This function uses a while loop to repeatedly ask the user for input until the user enters a valid option. This avoids the possibility of invalid inputs, such as numbers outside the given range or non-numeric entries, causing the program to crash or behave erratically. The input validation is effective, with a time complexity of O(1) at each check, and thus improves the robustness and usability of the program since it guides players to making valid selections.

The final loot is calculated using a greedy algorithm, which deducts the penalties for each phase of the heist, which are hacking, escape driving, and combat, based on the skills of the chosen crew. For instance, if the hacking skill is too low, then the amount of time available to loot the store is reduced; in the case of poor driving skills, losses during escape increase. The greedy approach was considered here because it is simple and effective for optimizing the loot output based on immediate decisions; thus, its time complexity at a calculation phase is of the order O(1) in this type of logic in games.

Direct access is used to retrieve crew members by player choice, in that although the program is not based on a certain searching algorithm, it actually can find the crew based upon a player's choices, and so it is efficient on small datasets because using indices provides constant time retrieval; in an extension of this program where more dynamic choices were included for the crews, a searching algorithm might be added to identify certain criteria, such as the most hack-able.