BITP 1123: DATA STRUCTURE & ALGORITHMS

Mini Project Report

Project Title: Peninsular Malaysia Weather Station(Temperature)

Team members:

Matric	Name	Sek/ Kump
B032310366	Ding Jia Jun	S1G1
B032310333	Chen Jin Han	S1G1
B032310317	Tan Yi Xin	S1G1
B032310371	Choong William	S1G1
B032310606	Eldhon Cheong Jie Qi	S1G1

Report Content

Report on Peninsular Malaysia Weather Station (Temperature)

1. Your Application

The "Peninsular Malaysia Weather Station" application is designed to manage and analyze temperature data for various states in Peninsular Malaysia. The application allows users to:

- Input temperature data (date, highest temperature, and lowest temperature) for different states.
- Search for temperature records by date or temperature.
- Sort the temperature data based on dates or temperature values.

The application includes:

- A main menu to select different states.
- Options to choose the search or sort method.
- Display of results based on user queries.

Explanation of Code

PeninsularMalaysia.h

This header file declares the p_Malaysia class, including its private data members (state, date, high temperature, low temperature) and public member functions.

PeninsularMalaysia.cpp

This file implements the member functions of the p_Malaysia class, including:

- Constructors and destructors.
- Functions to set data (setData), print data (printData), and get attributes like state, date, high temperature, and low temperature.

It also contains the implementation of search algorithms:

- SearchByDate: Uses binary search to find a record by date.
- SearchByTemp: Uses sequential search to find records by temperature.

Main.cpp

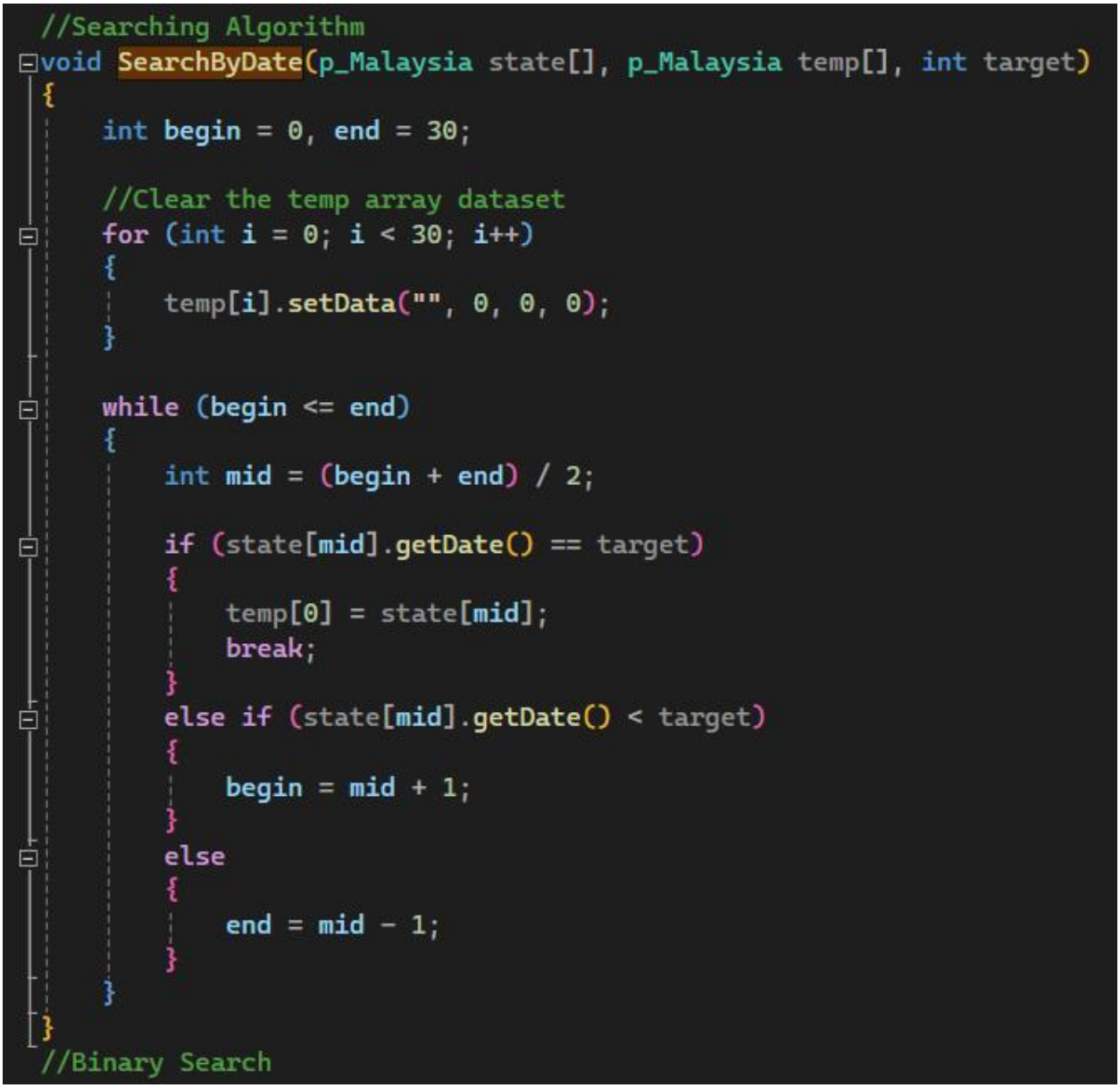
This file manages the user interface and controls the flow of the program. It presents a menu to the user, allows them to select a state and then choose a search or sort method. The code handles user inputs and calls appropriate functions to process the temperature data.

2. How the Chosen Searching and Sorting Algorithm Works

The application employs two main types of algorithms for data manipulation: searching and sorting.

Searching Algorithms

1. **Binary Search**:
    - **Purpose**: To find a temperature record by its date.
    - **How it works**:
        - The algorithm assumes the data is sorted by date.
        - It repeatedly divides the search interval in half.
        - It compares the target date with the date at the middle of the interval.
        - If the target date matches the middle date, the search is successful.
        - If the target date is less than the middle date, the search continues in the left half.
        - If the target date is greater, the search continues in the right half.
        - This process continues until the target date is found or the interval is empty.
    - **Advantages**: Efficient for large datasets with a time complexity of O(log n).



```
//Searching Algorithm
void SearchByDate(p_Malaysia state[], p_Malaysia temp[], int target)
{
    int begin = 0, end = 30;

    //Clear the temp array dataset
    for (int i = 0; i < 30; i++)
    {
        temp[i].setData("", 0, 0, 0);
    }

    while (begin <= end)
    {
        int mid = (begin + end) / 2;

        if (state[mid].getDate() == target)
        {
            temp[0] = state[mid];
            break;
        }
        else if (state[mid].getDate() < target)
        {
            begin = mid + 1;
        }
        else
        {
            end = mid - 1;
        }
    }
}
//Binary Search
```

2.1.1 Code snippet of binary search – searching algorithm of SearchByDate

How it Works:

- **Initialization**: The search starts with the whole array (begin = 0, end = 29).
- **Clearing temp array**: Before searching, it clears the temp array to store results.
- **Search Process**:
    - Calculate the mid-point: mid = (begin + end) / 2.

    - Compare the target date with the date at the mid-point (state[mid].getDate()).
    - If the target date matches, store the record in temp and break the loop.
    - If the target date is less than the mid-point date, narrow the search to the left half (end = mid - 1).
    - If the target date is greater, narrow the search to the right half (begin = mid + 1).
- **Time Complexity**: O(log n).

Example:

- Array of dates: [20230101, 20230102, 20230103, ..., 20230130].
- Target date: 20230115.
- Mid-point (initial): 20230115.
- If target is found at mid-point, search stops.
- If not, the search continues in the appropriate half of the array.

2. **Sequential Search**:
    - **Purpose**: To find temperature records by their temperature values.
    - **How it works**:
        - The algorithm iterates through each record in the dataset.
        - It checks if the highest or lowest temperature matches the target temperature.
        - If a match is found, it stores the record.
        - The search continues until all records are checked.
    - **Advantages**: Simple to implement and works well for small datasets, with a time complexity of O(n).

```
bool SearchByTemp(p_Malaysia state[], p_Malaysia temp[], int target)
{
    bool founded = false;

    //Clear the temp array dataset
    for (int i = 0; i < 30; i++)
    {
        temp[i].setData("", 0, 0, 0);
    }

    for (int i = 0; i < 30; i++)
    {
        if (state[i].getHighTemp() == target || state[i].getLowTemp() == target)
        {
            temp[i] = state[i];
            founded = true;
        }
    }

    return founded;
}
//Sequential Search
```

2.2.1 Code snippet of Sequential Search – searching algorithm of SearchByTemp

How it Works:

- **Initialization**: Clear the temp array to store results.
- **Search Process**:
    - Iterate through each record in the array.
    - Check if the highTemp or lowTemp matches the target temperature.
    - If a match is found, store the record in temp and set founded to true.
- **Time Complexity**: O(n).

Example:

- Array of temperatures: [{highTemp: 32, lowTemp: 25}, {highTemp: 30, lowTemp: 24}, ..., {highTemp: 33, lowTemp: 27}].
- Target temperature: 30.
- The search iterates through each record and stores the records with the matching temperature in temp.

Sorting Algorithms

The sorting algorithms used in the application are however not explicitly detailed in the provided code snippets, but typically such applications use common sorting techniques like:

1. **Bubble Sort**:
    - **How it works**:
        - It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
        - This process repeats until the list is sorted.
    - **Advantages**: Simple to implement, but inefficient for large datasets with a time complexity of O(n^2).
2. **Insertion Sort**:
    - **How it works**:
        - It builds the sorted array one item at a time.
        - It picks the next item and scans backwards through the sorted section to find the correct position.
        - The item is then inserted into its correct position.
    - **Advantages**: More efficient than bubble sort for small datasets, with a time complexity of O(n^2) in the worst case and O(n) in the best case.

```
void SortDateAsAscOrd(p_Malaysia state[], p_Malaysia temp[])
{
    for (int i = 0; i < 30; i++)
    {
        temp[i] = state[i];
    }

}
```

2.2.1 Snippet code of sorting algorithm - SortDateAsAscOrd

Explanation:

- This function copies the elements from the state array to the temp array.
- Currently, it does not perform any sorting after copying.

Example:

- Initial array of dates: [20230103, 20230101, 20230102].
- After copying: temp = [20230103, 20230101, 20230102].

```
void SortDateAsDescOrd(p_Malaysia state[], p_Malaysia temp[])
{
    int i, j;
    p_Malaysia key[1];

    for (i = 0; i < 30; i++)
    {
        temp[i] = state[i];
    }

    for (i = 1; i < 30;i++)
    {
        key[0] = temp[i];
        j = i - 1;

        while(j>=0 && temp[j].getDate() < key[0].getDate())
        {
            temp[j + 1] = temp[j];
            j -= 1;
        }
        temp[j + 1] = key[0];
    }

}
```

2.2.2 Snippet code of sorting algorithm - SortDateAsDescOrd

Explanation:

- **Initialization**: Copies the state array to the temp array.
- **Insertion Sort**:
    - Starts from the second element (i = 1).
    - Compares the current element with the elements before it.
    - Shifts elements to the right until the correct position for the current element is found.
    - Inserts the current element at the correct position.
- **Sorts Dates in Descending Order**.

Example:

- Initial array of dates: [20230103, 20230101, 20230102].
- After copying: temp = [20230103, 20230101, 20230102].
- Sorted array: [20230103, 20230102, 20230101].

```
void SortLowTempAscOrd(p_Malaysia state[], p_Malaysia temp[])
{
        int i, j;
        p_Malaysia key[1];

        for (i = 0; i < 30; i++)
        {
            temp[i] = state[i];
        }


        for (i = 1; i < 30;i++)
        {
            key[0] = temp[i];
            j = i - 1;

            while (j >= 0 && temp[j].getLowTemp() > key[0].getLowTemp() && temp[j].getLowTemp() != 0)
            {
                temp[j + 1] = temp[j];
                j -= 1;
            }
            temp[j + 1] = key[0];
        }
}
```

2.2.3 Snippet code of sorting algorithm - SortLowTempAscOrd

Explanation:

- **Initialization**: Copies the state array to the temp array.
- **Insertion Sort**:
    - Starts from the second element (i = 1).
    - Compares the current element's low temperature with the elements before it.
    - Shifts elements to the right until the correct position for the current element is found.
    - Inserts the current element at the correct position.
- **Sorts Low Temperatures in Ascending Order**.

Example:

- Initial array of low temperatures: [25, 23, 24].
- After copying: temp = [25, 23, 24].
- Sorted array: [23, 24, 25].

```
void SortLowTempDescOrd(p_Malaysia state[], p_Malaysia temp[])
{
    int i, j;
    p_Malaysia key[1];

    for (i = 0; i < 30; i++)
    {
        temp[i] = state[i];
    }


    for (i = 1; i < 30;i++)
    {
        key[0] = temp[i];
        j = i - 1;

        while (j >= 0 && temp[j].getLowTemp() < key[0].getLowTemp() && temp[j].getLowTemp() != 0)
        {
            temp[j + 1] = temp[j];
            j -= 1;
        }
        temp[j + 1] = key[0];
    }
}
```

2.2.3 Snippet code of sorting algorithm - SortLowTempDescOrd

Explanation:

- **Initialization**: Copies the state array to the temp array.
- **Insertion Sort**:
    - Starts from the second element (i = 1).
    - Compares the current element's low temperature with the elements before it.
    - Shifts elements to the right until the correct position for the current element is found.
    - Inserts the current element at the correct position.
- **Sorts Low Temperatures in Descending Order**.

Example:

- Initial array of low temperatures: [25, 23, 24].

- After copying: temp = [25, 23, 24].
- Sorted array: [25, 24, 23].

```
void SortHighTempAscOrd(p_Malaysia state[], p_Malaysia temp[])
{
    int i, j;
    p_Malaysia key[1];

    for (i = 0; i < 30; i++)
    {
        temp[i] = state[i];
    }


    for (i = 1; i < 30;i++)
    {
        key[0] = temp[i];
        j = i - 1;

        while (j >= 0 && temp[j].getHighTemp() > key[0].getHighTemp() && temp[j].getHighTemp() != 0)
        {
            temp[j + 1] = temp[j];
            j -= 1;
        }
        temp[j + 1] = key[0];
    }
}
```

2.2.3 Snippet code of sorting algorithm - SortHighTempAscOrd

Explanation:

- **Initialization**: Copies the state array to the temp array.
- **Insertion Sort**:
    - Starts from the second element (i = 1).
    - Compares the current element's high temperature with the elements before it.
    - Shifts elements to the right until the correct position for the current element is found.
    - Inserts the current element at the correct position.
- **Sorts High Temperatures in Ascending Order**.

Example:

- Initial array of high temperatures: [32, 30, 31].
- After copying: temp = [32, 30, 31].

- Sorted array: [30, 31, 32].

```
void SortHighTempDescOrd(p_Malaysia state[], p_Malaysia temp[])
{
    int i, j;
    p_Malaysia key[1];

    for (i = 0; i < 30; i++)
    {
        temp[i] = state[i];
    }


    for (i = 1; i < 30;i++)
    {
        key[0] = temp[i];
        j = i - 1;

        while (j >= 0 && temp[j].getHighTemp() < key[0].getHighTemp() && temp[j].getHighTemp() != 0)
        {
            temp[j + 1] = temp[j];
            j -= 1;
        }
        temp[j + 1] = key[0];
    }
}
```

2.2.3 Snippet code of sorting algorithm - SortHighTempDescOrd

Explanation:

- **Initialization**: Copies the state array to the temp array.
- **Insertion Sort**:
    - Starts from the second element (i = 1).
    - Compares the current element's high temperature with the elements before it.
    - Shifts elements to the right until the correct position for the current element is found.
    - Inserts the current element at the correct position.
- **Sorts High Temperatures in Descending Order**.

Example:

- Initial array of high temperatures: [32, 30, 31].
- After copying: temp = [32, 30, 31].
- Sorted array: [32, 31, 30].

Conclusion

The Peninsular Malaysia Weather Station application is a straightforward and efficient tool for managing temperature data. By using binary and sequential search algorithms, it ensures quick and reliable data retrieval. The choice of sorting algorithms, though not explicitly stated, is crucial for maintaining data order and improving search efficiency.

C++ Codes

Header file – PeninsularMalaysia.h

```
#ifndef PENINSULARMALAYSIA_H
#define PENINSULARMALAYSIA_H

#include <iostream>

using namespace std;

class p_Malaysia
{
	private:
		string state;
		int date, highTemp, lowTemp;

	public:
		p_Malaysia();
		~p_Malaysia();
		void setData(string state, int date, int highTemp, int lowTemp);
		void printData();
		string getState();
		int getDate();
		int getHighTemp();
		int getLowTemp();
};

#endif
```

CPP file – PeninsularMalaysia.cpp

```
#include <iostream>
#include "PeninsularMalaysia.h"

using namespace std;

p_Malaysia::p_Malaysia()
{
	date = 0;
	highTemp = 0;
	lowTemp = 0;
}


p_Malaysia::~p_Malaysia() {}

void p_Malaysia::setData(string state,int date, int highTemp, int lowTemp)
{
	this->state = state;
	this->date = date;
	this->highTemp = highTemp;
	this->lowTemp = lowTemp;
}

void p_Malaysia::printData()
{
	cout << "| Date: " << date << "\t| Highest Temp: " << highTemp
		<< " \t| Lowest Temp: " << lowTemp << "\t | " << endl;
}

string p_Malaysia::getState()
{
	return state;
}

int p_Malaysia::getDate()
{
	return date;
}

int p_Malaysia::getHighTemp()
{
	return highTemp;
}

int p_Malaysia::getLowTemp()
{
	return lowTemp;
}
```

CPP file - Main.cpp

```
#include <iostream>
#include <string>
#include "PeninsularMalaysia.h"

using namespace std;

//DataSet
void dataSet(p_Malaysia[], p_Malaysia[], p_Malaysia[], p_Malaysia[], p_Malaysia[],
p_Malaysia[], p_Malaysia[],
                p_Malaysia[], p_Malaysia[], p_Malaysia[], p_Malaysia[], p_Malaysia[],
p_Malaysia[]);

//Menu Function
void StartMenu();
void AlgorithmMenu(p_Malaysia[], int&, bool&);
bool MethodMenu(p_Malaysia[], p_Malaysia[], int, bool);
void PrintReport(p_Malaysia temp[]);
void CinClear();

//Sorting Algorithm (Insertion Sort)
void SortDateAsAscOrd(p_Malaysia state[], p_Malaysia temp[]);
void SortDateAsDescOrd(p_Malaysia state[], p_Malaysia temp[]);
void SortLowTempAscOrd(p_Malaysia state[], p_Malaysia temp[]);
void SortLowTempDescOrd(p_Malaysia state[], p_Malaysia temp[]);
void SortHighTempAscOrd(p_Malaysia state[], p_Malaysia temp[]);
void SortHighTempDescOrd(p_Malaysia state[], p_Malaysia temp[]);

//Searching Algorithm
void SearchByDate(p_Malaysia state[], p_Malaysia temp[], int target);
bool SearchByTemp(p_Malaysia state[], p_Malaysia temp[], int target);

//Main Function
int main()
{
	//Data
	p_Malaysia Johor[30], Kedah[30], Kelantan[30], Kuala_Lumpur[30], Malacca[30],
Negeri_Sembilan[30], Pahang[30],
		Penang[30], Perak[30], Perlis[30], Putrajaya[30], Selangor[30],
Terengganu[30], temp[30];

	dataSet(Johor, Kedah, Kelantan, Kuala_Lumpur, Malacca, Negeri_Sembilan,
Pahang, Penang, Perak, Perlis, Putrajaya, Selangor, Terengganu);

	int choice = 0, method = 0;
	bool valid = false, exit = false, back = false;

	do
	{
		exit = false, valid = false, back = false;
		//Detect user's choice on choosing state
		do
		{
			StartMenu();
			cin >> choice;
```

```
                    //Detect if theres input that are not integer, it will reask the
user's input
                    if (cin.fail())
                    {
                          CinClear();
                          cout << "Please select according the menu !";
                          continue;
                    }

                    //Detect the user's input is between 0 - 13, if not it will
reask the user to input again
                    if (choice >= 0 && choice <= 13)
                    {
                          choice == 0 ? exit = true : exit = false;
                          valid = true;
                    }
                    else
                    {
                          system("cls");
                          cout << "Please select according the menu !";

                    }
              } while (valid == false);

              system("cls");

              //Going to the state's report option
              while (back == false && choice != 0)
              {
                    method = 0;

                    switch (choice)
                    {
                          case 1:
                                AlgorithmMenu(Johor, method, back);
                                if (MethodMenu(Johor, temp, method, back))
{ continue; };
                                break;

                          case 2:
                                AlgorithmMenu(Kedah, method, back);
                                if (MethodMenu(Kedah, temp, method, back))
{ continue; };
                                break;

                          case 3:
                                AlgorithmMenu(Kelantan, method, back);
                                if (MethodMenu(Kelantan, temp, method, back))
{ continue; };
                                break;

                          case 4:
                                AlgorithmMenu(Kuala_Lumpur, method, back);
                                if (MethodMenu(Kuala_Lumpur, temp, method, back))
{ continue; };
                                break;

                          case 5:
```

```
                                    AlgorithmMenu(Malacca, method, back);
                                    if (MethodMenu(Malacca, temp, method, back))
{ continue; };
                                    break;

                              case 6:
                                    AlgorithmMenu(Negeri_Sembilan, method, back);
                                    if (MethodMenu(Negeri_Sembilan, temp, method, back))
{ continue; };
                                    break;

                              case 7:
                                    AlgorithmMenu(Pahang, method, back);
                                    if (MethodMenu(Pahang, temp, method, back))
{ continue; };
                                    break;

                              case 8:
                                    AlgorithmMenu(Penang, method, back);
                                    if (MethodMenu(Penang, temp, method, back))
{ continue; };
                                    break;

                              case 9:
                                    AlgorithmMenu(Perak, method, back);
                                    if (MethodMenu(Perak, temp, method, back))
{ continue; };
                                    break;

                              case 10:
                                    AlgorithmMenu(Perlis, method, back);
                                    if (MethodMenu(Perlis, temp, method, back))
{ continue; };
                                    break;

                              case 11:
                                    AlgorithmMenu(Putrajaya, method, back);
                                    if (MethodMenu(Putrajaya, temp, method, back))
{ continue; };
                                    break;

                              case 12:
                                    AlgorithmMenu(Selangor, method, back);
                                    if (MethodMenu(Selangor, temp, method, back))
{ continue; };
                                    break;

                              case 13:
                                    AlgorithmMenu(Terengganu, method, back);
                                    if (MethodMenu(Terengganu, temp, method, back))
{ continue; };
                                    break;
                        }

                  }

            } while (exit == false);
```

```
	cout << "Thank you for using our service!\n\n\n\n";


	return 0;
}

//DataSet
void dataSet(p_Malaysia Johor[], p_Malaysia Kedah[], p_Malaysia Kelantan[],
p_Malaysia Kuala_Lumpur[], p_Malaysia Malacca[], p_Malaysia Negeri_Sembilan[],
		p_Malaysia Pahang[],p_Malaysia Penang[], p_Malaysia Perak[],
p_Malaysia Perlis[], p_Malaysia Putrajaya[], p_Malaysia Selangor[],
		p_Malaysia Terengganu[])
{
	//Johor
	int Johor_high_tem[] = { 33, 34, 34, 36, 35, 35, 36, 33, 34, 32, 33, 27, 34,
32, 33, 31, 33, 33, 33, 32, 35, 35, 33, 32, 34, 36, 35, 34, 35, 35 };
	int Johor_low_tem[] = { 25,26, 26, 24, 25, 25, 26, 27, 25, 26, 26, 25, 24 ,26,
26, 26, 25, 25, 26, 26, 25, 26, 25, 25, 26, 25, 26, 26, 25, 26 };
	for (int i = 0; i < 30;i++)
	{
		Johor[i].setData("Johor", 20240400 + i + 1, Johor_high_tem[i],
Johor_low_tem[i]);
	}

	//Kedah
	int Kedah_high_tem[] = { 33, 33, 34, 34, 34, 34, 35, 35, 35, 36, 37, 37, 36,
35, 34, 34, 33, 33, 33, 33, 34, 34, 33, 33, 33, 33, 33, 34, 33, 34 };
	int Kedah_low_tem[] = { 27, 28, 28, 28, 27, 27, 25, 27, 26, 28, 27, 28, 28,
28, 28, 27, 26, 28, 26, 27, 28, 28, 29, 27, 27, 27, 28, 27, 26, 26 };
	for (int i = 0; i < 30;i++)
	{
		Kedah[i].setData("Kedah", 20240400 + i + 1, Kedah_high_tem[i],
Kedah_low_tem[i]);
	}

	//Kelantan
	int Kelantan_high_tem[] = { 33, 34, 34, 33, 36, 34, 33, 34, 34, 33, 33, 33,
33, 34, 34, 34, 34, 33, 33, 34, 34, 35, 35, 35, 35, 35, 35, 36, 35, 34 };
	int Kelantan_low_tem[] = { 26, 26, 27, 25, 25, 24, 24, 25, 27, 27, 26, 26, 26,
26, 26, 27, 26, 26, 26, 26, 26, 27, 27, 27, 27, 26, 26, 28, 27, 27 };
	for (int i = 0; i < 30;i++)
	{
		Kelantan[i].setData("Kelantan", 20240400 + i + 1, Kelantan_high_tem[i],
Kelantan_low_tem[i]);
	}

	//Kuala Lumpur
	int Kuala_Lumpur_high_tem[] = { 34, 34, 35, 34, 35, 35, 35, 31, 34, 32, 35,
35, 35, 35, 35, 35, 33, 33, 33, 34, 35, 35, 34, 34, 35, 35, 34, 34, 36, 35 };
	int Kuala_Lumpur_low_tem[] = { 26, 26, 27, 27, 27, 27, 26, 26, 26, 26, 25, 26,
26, 26, 26, 24, 24, 26, 26, 27, 27, 26, 26, 26, 25, 26, 26, 26, 26, 26 };
	for (int i = 0; i < 30;i++)
	{
		Kuala_Lumpur[i].setData("Kuala Lumpur", 20240400 + i + 1,
Kuala_Lumpur_high_tem[i], Kuala_Lumpur_low_tem[i]);
	}

	//Malacca
```

```
	int Malacca_high_tem[] = { 34, 34, 33, 34, 34, 34, 34, 32, 34, 32, 33, 32, 35,
34, 34, 33, 31, 33, 33, 32, 32, 33, 33, 32, 33, 33, 33, 32, 33, 34 };
	int Malacca_low_tem[] = { 26, 26, 27, 27, 27, 27, 26, 26, 26, 26, 25, 26, 26,
26, 26, 24, 24, 26, 26, 27, 27, 26, 26, 26, 25, 26, 26, 26, 26, 26 };
	for (int i = 0; i < 30;i++)
	{
		Malacca[i].setData("Malacca", 20240400 + i + 1, Malacca_high_tem[i],
Malacca_low_tem[i]);
	}

	//Negeri Sembilan
	int Negeri_Sembilan_high_tem[] = { 35, 34, 33, 34, 34, 34, 34, 31, 35, 34, 35,
35, 36, 35, 35, 34, 31, 33, 33, 33, 34, 34, 34, 33, 34, 34, 34, 34, 34, 34 };
	int Negeri_Sembilan_low_tem[] = { 25, 25, 25, 26, 25, 25, 26, 26, 25, 24, 25,
24, 26, 25, 26, 25, 25, 26, 26, 27, 27, 26, 27, 26, 26, 26, 26, 25, 26, 27 };
	for (int i = 0; i < 30;i++)
	{
		Negeri_Sembilan[i].setData("Negeri Sembilan", 20240400 + i + 1,
Negeri_Sembilan_high_tem[i], Negeri_Sembilan_low_tem[i]);
	}

	//Pahang
	int Pahang_high_tem[] = { 35, 35, 34, 35, 35, 35, 35, 35, 35, 31, 34, 34, 31,
32, 30, 31, 33, 33, 35, 33, 35, 34, 34, 34, 34, 34, 34, 35, 35, 35 };
	int Pahang_low_tem[] = { 25, 25, 27, 26, 24, 24, 24, 25, 26, 25, 25, 25, 26,
25, 25, 26, 24, 24, 24, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 27 };
	for (int i = 0; i < 30;i++)
	{
		Pahang[i].setData("Pahang", 20240400 + i + 1, Pahang_high_tem[i],
Pahang_low_tem[i]);
	}

	//Penang
	int Penang_high_tem[] = { 32, 33, 33, 33, 33, 34, 33, 34, 33, 33, 34, 34, 34,
34, 34, 33, 32, 33, 33, 33, 33, 34, 33, 34, 34, 33, 34, 35, 35, 34 };
	int Penang_low_tem[] = { 25, 26, 26, 27, 26, 27, 26, 26, 26, 25, 24, 26, 27,
25, 27, 27, 27, 27, 26, 27, 28, 27, 27, 26, 26, 26, 26, 27, 26, 27 };
	for (int i = 0; i < 30;i++)
	{
		Penang[i].setData("Penang", 20240400 + i + 1, Penang_high_tem[i],
Penang_low_tem[i]);
	}

	//Perak
	int Perak_high_tem[] = { 33, 34, 34, 35, 35, 36, 35, 33, 34, 34, 35, 35, 34,
33, 34, 33, 33, 34, 33, 34, 36, 34, 33, 34, 34, 34, 34, 33, 34, 34 };
	int Perak_low_tem[] = { 25, 26, 26, 26, 25, 25, 26, 27, 25, 25, 24, 25, 27,
23, 24, 26, 25, 26, 25, 26, 25, 25, 25, 25, 26, 26, 26, 26, 27, 26 };
	for (int i = 0; i < 30;i++)
	{
		Perak[i].setData("Perak", 20240400 + i + 1, Perak_high_tem[i],
Perak_low_tem[i]);
	}

	//Perlis
	int Perlis_high_tem[] = { 34, 35, 36, 36, 36, 37, 37, 36, 36, 34, 37, 37, 37,
36, 37, 36, 34, 34, 34, 34, 34, 35, 35, 35, 35, 35, 36, 34, 35, 34 };
```

```
	int Perlis_low_tem[] = { 26, 25, 25, 25, 25, 25, 22, 26, 25, 26, 25, 25, 25,
26, 27, 25, 26, 26, 26, 25, 26, 27, 27, 25, 26, 26, 26, 27,26, 27 };
	for (int i = 0; i < 30;i++)
	{
		Perlis[i].setData("Perlis", 20240400 + i + 1, Perlis_high_tem[i],
Perlis_low_tem[i]);
	}

	//Putrajaya
	int Putrajaya_high_tem[] = { 35, 34, 33, 34, 34, 34, 34, 31, 35, 34, 35, 35,
36, 35, 35, 34, 31, 33, 33, 33, 33, 34, 34, 33, 34, 34, 34, 33, 34, 34 };
	int Putrajaya_low_tem[] = { 25, 26, 25, 26, 26, 26, 26, 26, 25, 26, 25, 24,
26, 26, 25, 25, 25, 26, 26, 26, 25, 26, 27, 27, 26, 26, 26, 25 ,26, 27 };
	for (int i = 0; i < 30;i++)
	{
		Putrajaya[i].setData("Putrajaya", 20240400 + i + 1,
Putrajaya_high_tem[i], Putrajaya_low_tem[i]);
	}


	//Selangor
	int Selangor_high_tem[] = { 34, 34, 35, 34, 35, 36, 35, 31, 34, 32, 34, 35,
35, 35, 35, 35, 33, 34, 34, 35, 36, 35, 34, 34, 35, 35, 35, 33, 36, 34 };
	int Selangor_low_tem[] = { 25, 25, 27, 26, 27, 27, 26, 25, 26, 26, 25, 26, 26,
26, 26, 24, 24, 24, 26, 27, 27, 27, 26, 26, 25, 26, 26, 26 ,27, 26 };
	for (int i = 0; i < 30;i++)
	{
		Selangor[i].setData("Selangor", 20240400 + i + 1, Selangor_high_tem[i],
Selangor_low_tem[i]);
	}

	//Terengganu
	int Terengganu_high_tem[] = { 34, 34, 34, 34, 33, 34, 33, 34, 34, 34, 34, 34,
34, 33, 34, 34, 34, 34, 34, 34, 34, 35, 35, 34, 35, 35, 35, 35, 35, 35 };
	int Terengganu_low_tem[] = { 26, 26, 26, 25, 25, 24, 24, 25, 27, 27, 26, 27,
26, 26, 26, 27, 26, 26, 26, 27, 26, 27, 27, 27, 27, 27, 27, 27 ,28, 27 };
	for (int i = 0; i < 30;i++)
	{
		Terengganu[i].setData("Terengganu", 20240400 + i + 1,
Terengganu_high_tem[i], Terengganu_low_tem[i]);
	}
}

//Menu Function
void StartMenu()
{
	cout <<
"\n\n=================================================================\n"
		<< "  Welcome to Peninsular Malaysia Weather Station for April 2024!
\n"
		<<
"=================================================================\n"
		<< "Please select a state: \n"
		<< "1. Johor\n"
		<< "2. Kedah\n"
		<< "3. Kelantan\n"
		<< "4. Kuala Lumpur\n"
		<< "5. Malacca\n"
```

```
		<< "6. Negeri Sembilan\n"
		<< "7. Pahang\n"
		<< "8. Penang\n"
		<< "9. Perak\n"
		<< "10. Perlis\n"
		<< "11. Putrajaya\n"
		<< "12. Selangor\n"
		<< "13. Terengganu\n\n\n\n"
		<< "0. Exit\n\n"
		<< "Option: ";
}

void AlgorithmMenu(p_Malaysia state[], int& method, bool& back)
{
	system("cls");
	bool valid = false;
	do
	{
		cout << "\n\n================================================"
			<< "\nCurrent state choosen: " << state[0].getState()
			<< "\n================================================"
			<< "\nPlease select the algorithm used on the report: "
			<< "\n1. Sorting"
			<< "\n2. Search"
			<< "\n\n\n\n0. Back\n\n"
			<< "Option: ";

		cin >> method;

		if (cin.fail())
		{
			CinClear();
			cout << "Please select according the menu !";
			continue;
		}

		if (method >= 0 && method <= 2)
		{
			valid = true;
		}
		else
		{
			system("cls");
			cout << "Please select according the menu !";

		}

		method == 0 ? back = true : back = false;
		system("cls");

	} while (valid == false);

}

bool MethodMenu(p_Malaysia state[], p_Malaysia temp[], int method, bool back)
{
	int hold = method;
	int target = 0;
```

```
	while (back == false)
	{
		method = hold;
		if (method == 1)
		{
			cout << "\n\n================================================"
				<< "\nCurrent state choosen: " << state[0].getState()
				<< "\n================================================"
				<< "\nPlease select the algorithm's method used on the
report: "
				<< "\n1. Sort by date in ascending order"
				<< "\n2. Sort by date in descending order"
				<< "\n3. Sort by highest temperature in ascending order"
				<< "\n4. Sort by highest temeperature in descending order"
				<< "\n5. Sort by lowest temperature in ascending order"
				<< "\n6. Sort by lowest temperature in descending order"
				<< "\n\n\n\n0. Back\n\n"
				<< "Option: ";

			cin >> method;

			if (cin.fail())
			{
				CinClear();
				cout << "Please select according the menu !";
				continue;
			}

			if (method == 0) { return back = true; }
			else if (method == 1) { SortDateAsAscOrd(state, temp);
PrintReport(temp);break; }
			else if (method == 2) { SortDateAsDescOrd(state, temp);
PrintReport(temp);break; }
			else if (method == 3) { SortHighTempAscOrd(state, temp);
PrintReport(temp);break; }
			else if (method == 4) { SortHighTempDescOrd(state, temp);
PrintReport(temp);break; }
			else if (method == 5) { SortLowTempAscOrd(state, temp);
PrintReport(temp);break; }
			else if (method == 6) { SortLowTempDescOrd(state, temp);
PrintReport(temp); break; }
			else
			{
				system("cls");
				cout << "Please select according the menu !";
			}

		}
		else
		{
			cout << "\n\n================================================"
				<< "\nCurrent state choosen: " << state[0].getState()
				<< "\n================================================"
				<< "\nPlease select the algorithm's method used on the
report: "
				<< "\n1. Search by date"
				<< "\n2. Search on temperature"
				<< "\n\n\n\n0. Back\n\n"
```

```
		<< "Option: ";

	cin >> method;

	if (cin.fail())
	{
		CinClear();
		cout << "Please select according the menu !";
		continue;
	}

	if (method == 0) { return back = true; }
	else if (method == 1)
	{
		cout << "\n\nDate to search (example: 20240417) : ";
		cin >> target;

		if (cin.fail())
		{
			CinClear();
			cout << "Please select according the menu !";
			continue;
		}

		if (target >= 20240401 && target <= 20240430)
		{
			SearchByDate(state, temp, target);
			PrintReport(temp);
			break;

		}
		else
		{
			system("cls");
			cout << "Please enter the date with correct
format !";
			continue;
		}
	}
	else if (method == 2)
	{
		cout << "\n\nTemperature to search (In Celcius) : ";
		cin >> target;

		if (cin.fail())
		{
			CinClear();
			cout << "Please select according the menu !";
			continue;
		}

		if (SearchByTemp(state, temp, target))
		{
			PrintReport(temp);
			break;
		}
		else
		{
```

```
                                system("cls");
                                cout << "Couldn't found the temperature !";
                                continue;
                        }
                }
                else {
                        system("cls");
                        cout << "Please select according the menu !";
                }
        }

    }

    return back;
}

void PrintReport(p_Malaysia temp[])
{
    system("cls");

    string hold;

    //find the name of the state
    for (int i = 0; i < 30; i++)
    {
        if (temp[i].getDate() != 0)
        {
            hold = temp[0].getState();
        }
    }

    cout << "Data Sources: Weather.com\n"
        << "State: " << hold << "\t-- 01/04/2024 to 30/04/2024\ttemperature
unit: Celcius\n"
        <<
"-------------------------------------------------------------------------\n";


    for (int i = 0; i < 30; i++)
    {
        if (temp[i].getDate() != 0)
        {
            temp[i].printData();
        }
    }

    cout << "-------------------------------------------------------------------
------\n"
        << "\n\n\n";

    system("pause");
}

void CinClear()
{
    cin.clear();
    cin.ignore();
    system("cls");
```

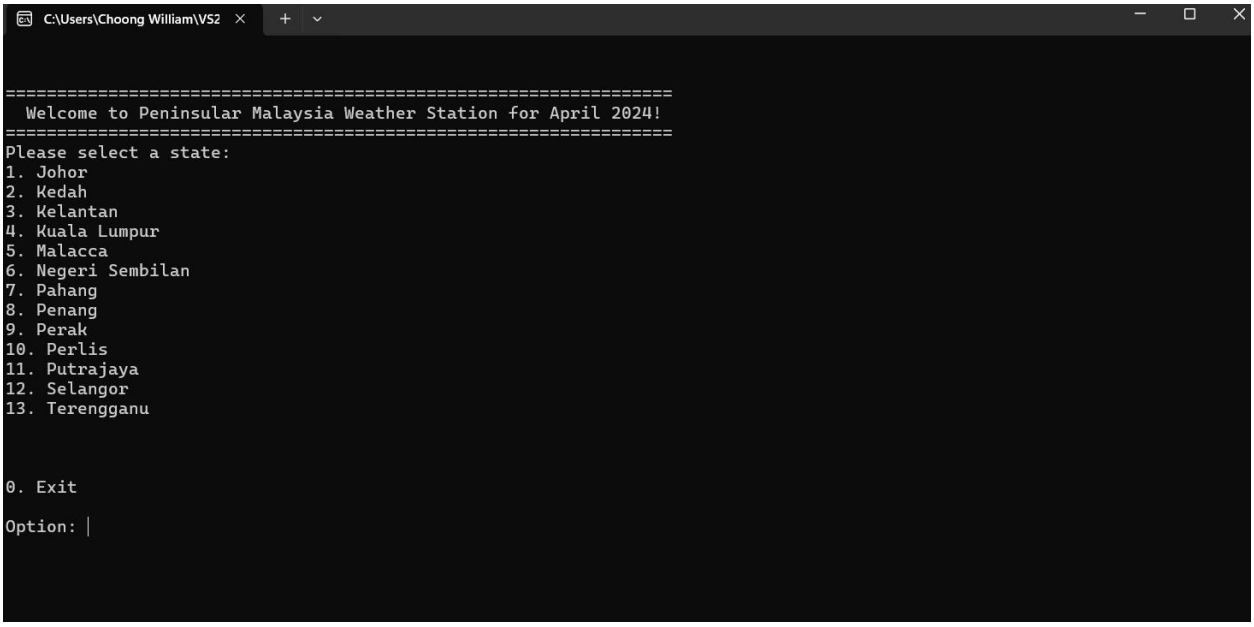
```
}


//Sorting Algorithm (Insertion Sort)
void SortDateAsAscOrd(p_Malaysia state[], p_Malaysia temp[])
{
	for (int i = 0; i < 30; i++)
	{
		temp[i] = state[i];
	}

}

void SortDateAsDescOrd(p_Malaysia state[], p_Malaysia temp[])
{
	int i, j;
	p_Malaysia key[1];

	for (i = 0; i < 30; i++)
	{
		temp[i] = state[i];
	}

	for (i = 1; i < 30;i++)
	{
		key[0] = temp[i];
		j = i - 1;

		while(j>=0 && temp[j].getDate() < key[0].getDate())
		{
			temp[j + 1] = temp[j];
			j -= 1;
		}
		temp[j + 1] = key[0];
	}

}

void SortLowTempAscOrd(p_Malaysia state[], p_Malaysia temp[])
{
		int i, j;
		p_Malaysia key[1];

		for (i = 0; i < 30; i++)
		{
			temp[i] = state[i];
		}


		for (i = 1; i < 30;i++)
		{
			key[0] = temp[i];
			j = i - 1;

			while (j >= 0 && temp[j].getLowTemp() > key[0].getLowTemp() &&
temp[j].getLowTemp() != 0)
			{
				temp[j + 1] = temp[j];
```

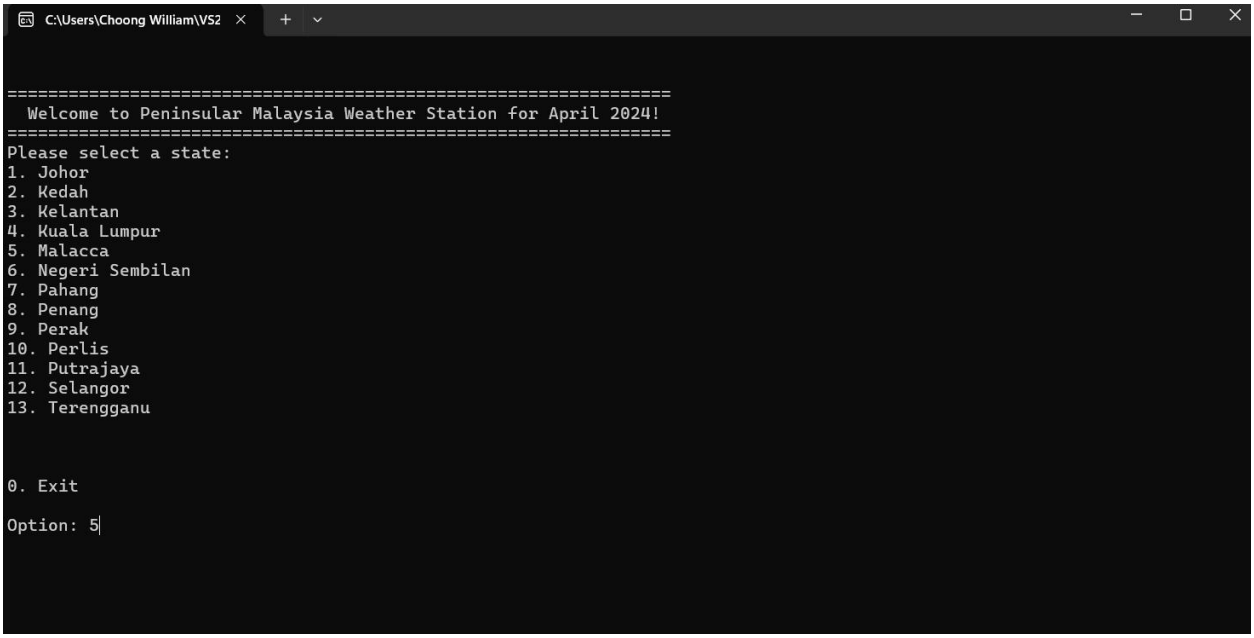
```
                        j -= 1;
                }
                temp[j + 1] = key[0];
        }
}

void SortLowTempDescOrd(p_Malaysia state[], p_Malaysia temp[])
{
        int i, j;
        p_Malaysia key[1];

        for (i = 0; i < 30; i++)
        {
                temp[i] = state[i];
        }


        for (i = 1; i < 30;i++)
        {
                key[0] = temp[i];
                j = i - 1;

                while (j >= 0 && temp[j].getLowTemp() < key[0].getLowTemp() &&
temp[j].getLowTemp() != 0)
                {
                        temp[j + 1] = temp[j];
                        j -= 1;
                }
                temp[j + 1] = key[0];
        }
}

void SortHighTempAscOrd(p_Malaysia state[], p_Malaysia temp[])
{
        int i, j;
        p_Malaysia key[1];

        for (i = 0; i < 30; i++)
        {
                temp[i] = state[i];
        }


        for (i = 1; i < 30;i++)
        {
                key[0] = temp[i];
                j = i - 1;

                while (j >= 0 && temp[j].getHighTemp() > key[0].getHighTemp() &&
temp[j].getHighTemp() != 0)
                {
                        temp[j + 1] = temp[j];
                        j -= 1;
                }
                temp[j + 1] = key[0];
        }
}
```

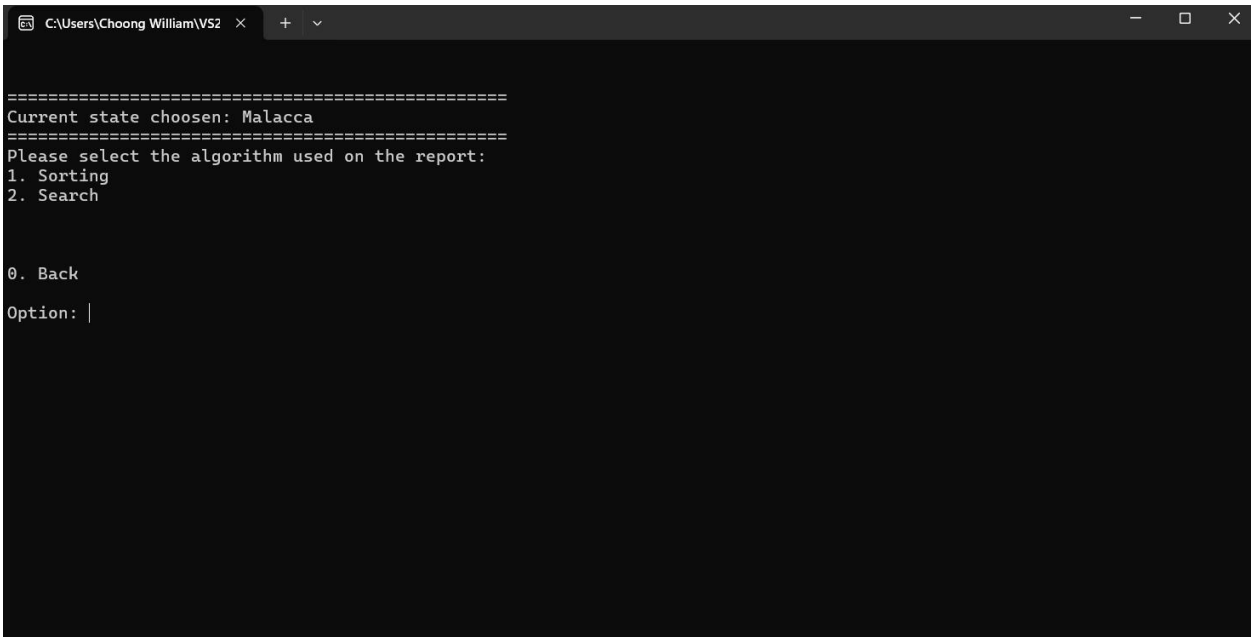
```
void SortHighTempDescOrd(p_Malaysia state[], p_Malaysia temp[])
{
	int i, j;
	p_Malaysia key[1];

	for (i = 0; i < 30; i++)
	{
		temp[i] = state[i];
	}


	for (i = 1; i < 30;i++)
	{
		key[0] = temp[i];
		j = i - 1;

		while (j >= 0 && temp[j].getHighTemp() < key[0].getHighTemp() &&
temp[j].getHighTemp() != 0)
		{
			temp[j + 1] = temp[j];
			j -= 1;
		}
		temp[j + 1] = key[0];
	}
}


//Searching Algorithm
void SearchByDate(p_Malaysia state[], p_Malaysia temp[], int target)
{
	int begin = 0, end = 30;

	//Clear the temp array dataset
	for (int i = 0; i < 30; i++)
	{
		temp[i].setData("", 0, 0, 0);
	}

	while (begin <= end)
	{
		int mid = (begin + end) / 2;

		if (state[mid].getDate() == target)
		{
			temp[0] = state[mid];
			break;
		}
		else if (state[mid].getDate() < target)
		{
			begin = mid + 1;
		}
		else
		{
			end = mid - 1;
		}
	}
}
//Binary Search
```

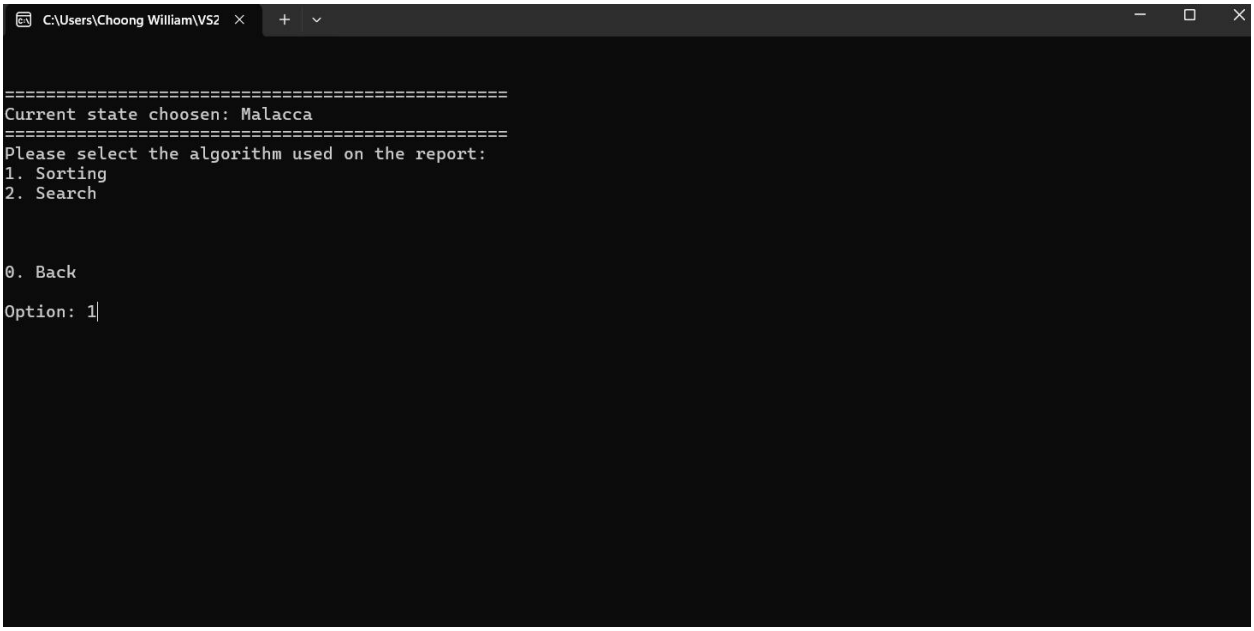
```
bool SearchByTemp(p_Malaysia state[], p_Malaysia temp[], int target)
{
	bool founded = false;

	//Clear the temp array dataset
	for (int i = 0; i < 30; i++)
	{
		temp[i].setData("", 0, 0, 0);
	}

	for (int i = 0; i < 30; i++)
	{
		if (state[i].getHighTemp() == target || state[i].getLowTemp() == target)
		{
			temp[i] = state[i];
			founded = true;
		}
	}

	return founded;
}
//Sequential Search
```

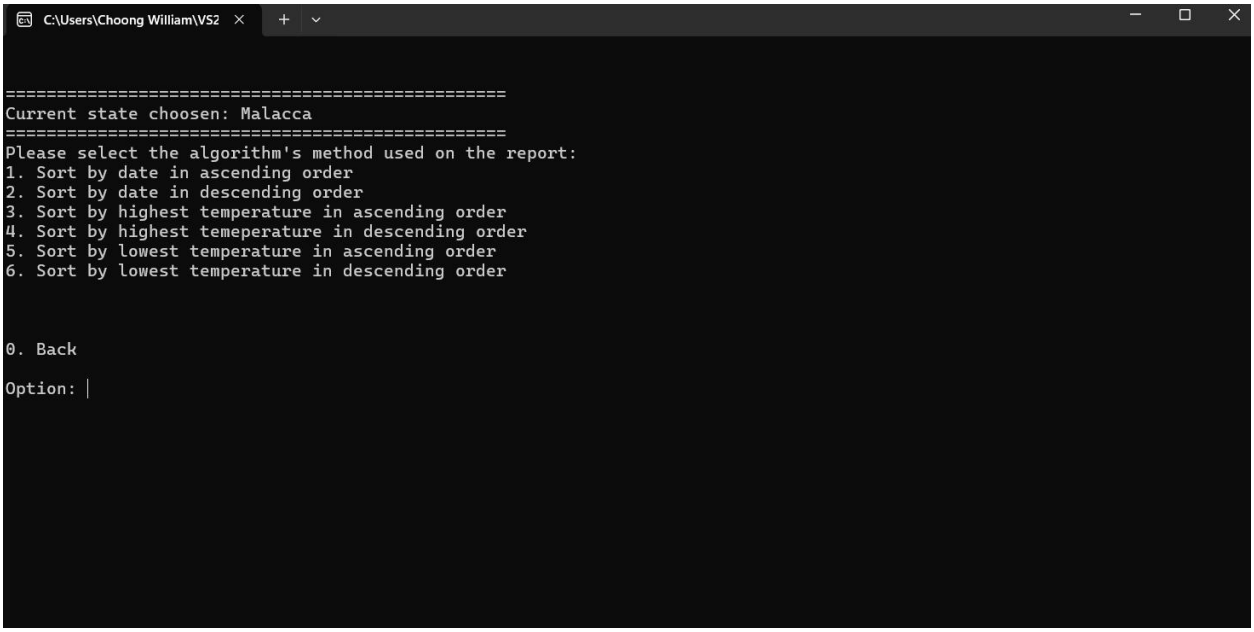
Output of Code

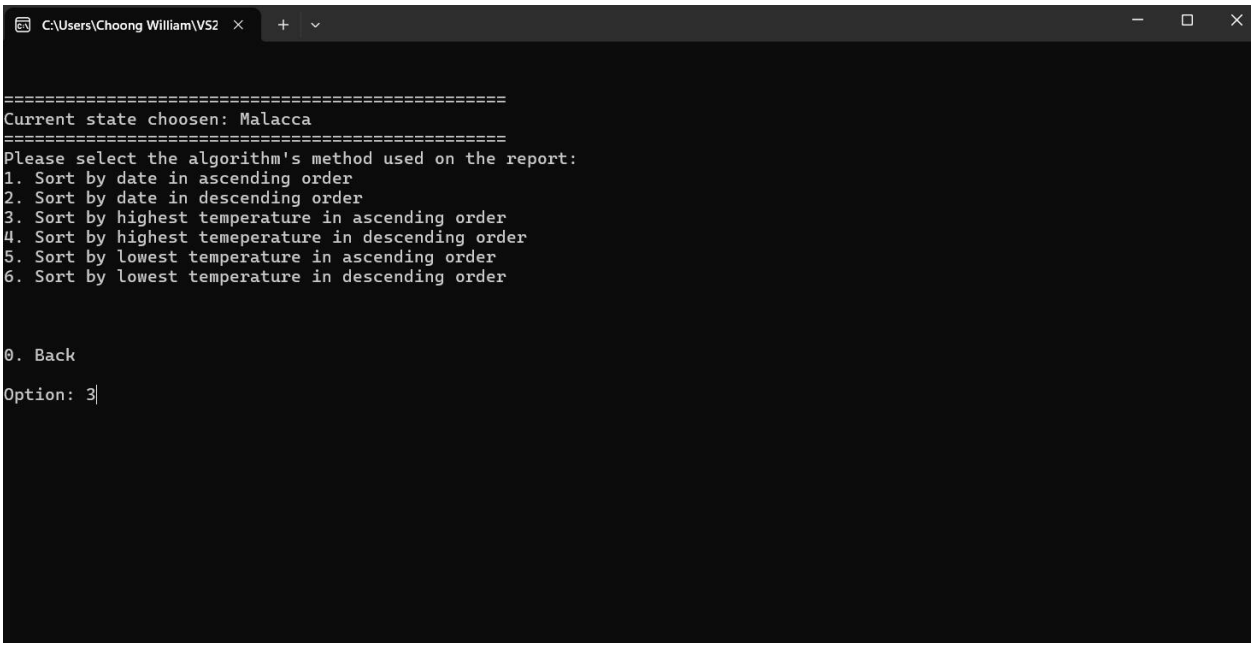


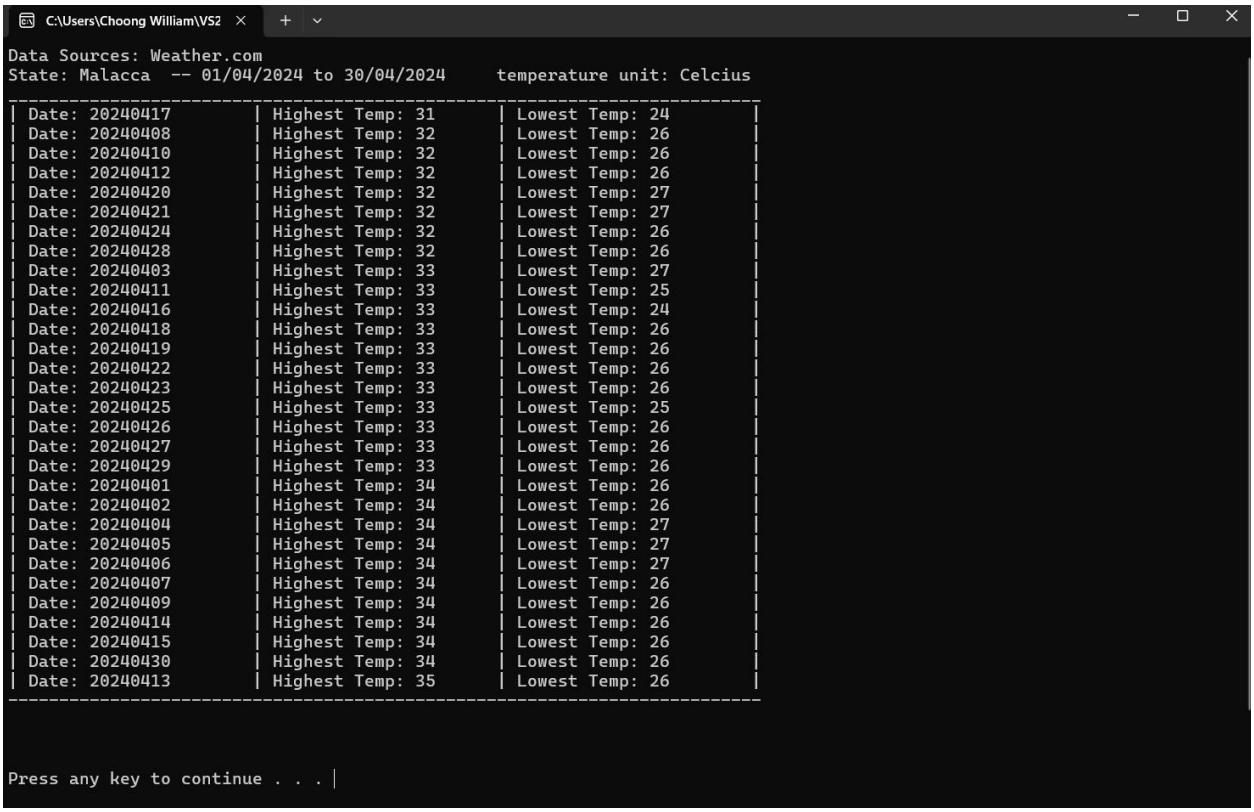


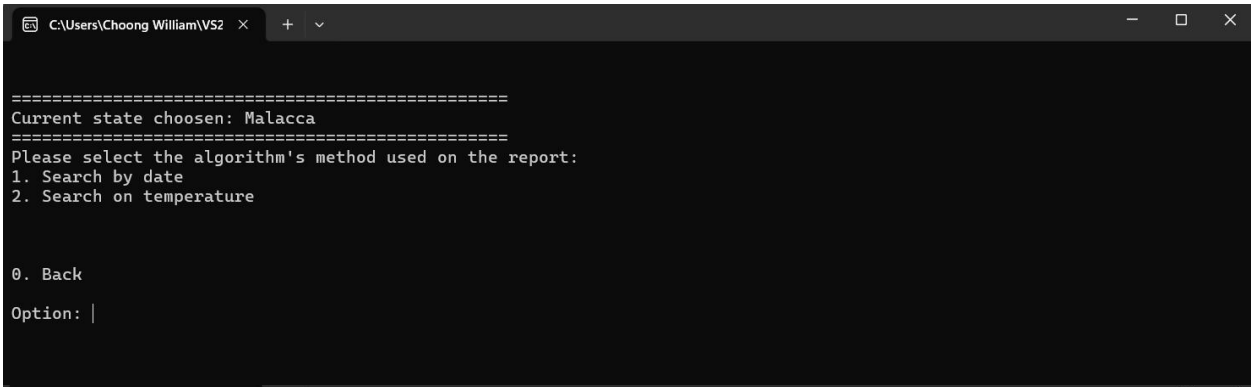

C:\Users\Choong William\VS2
==========
Current state choosen: Malacca
==========
Please select the algorithm used on the report:
1. Sorting
2. Search
0. Back
Option:

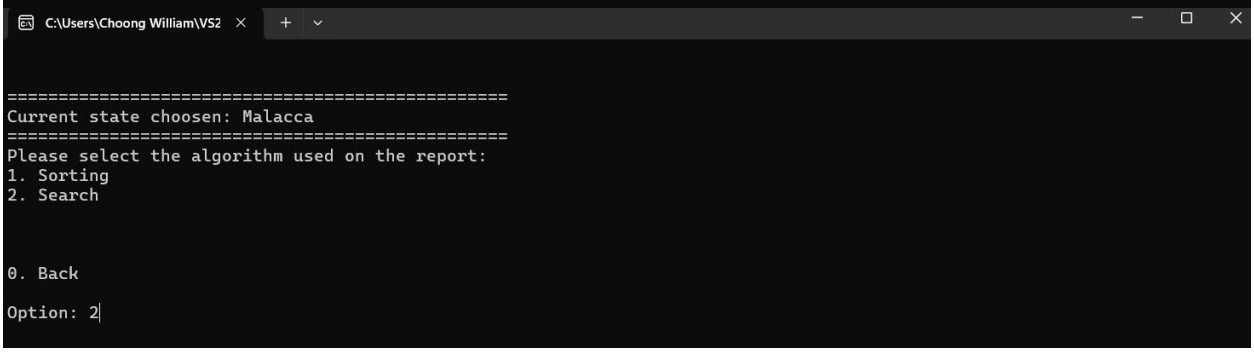

C:\Users\Choong William\VS2
==========
Current state choosen: Malacca
==========
Please select the algorithm used on the report:
1. Sorting
2. Search
0. Back
Option: 1

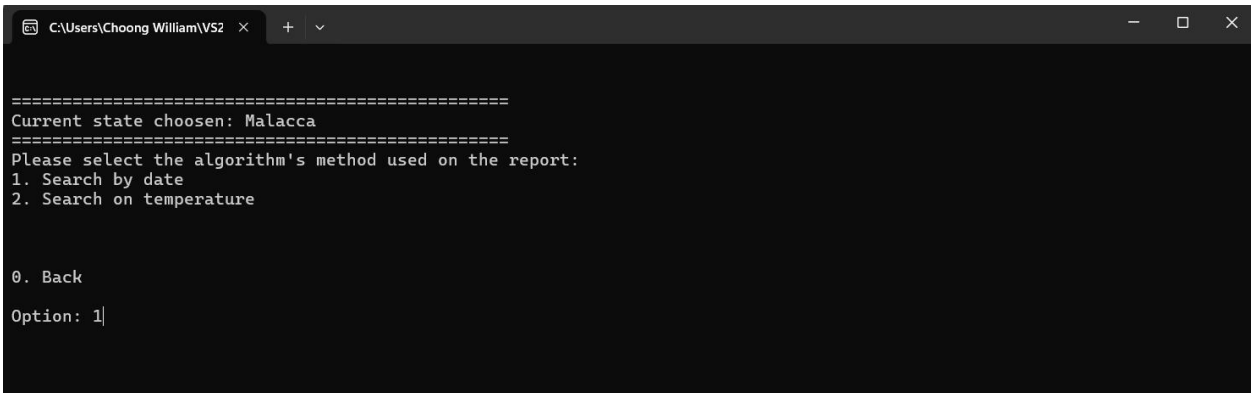

C:\Users\Choong William\VS2
==
Current state choosen: Malacca
==
Please select the algorithm's method used on the report:
1. Sort by date in ascending order
2. Sort by date in descending order
3. Sort by highest temperature in ascending order
4. Sort by highest temeperature in descending order
5. Sort by lowest temperature in ascending order
6. Sort by lowest temperature in descending order
0. Back
Option:

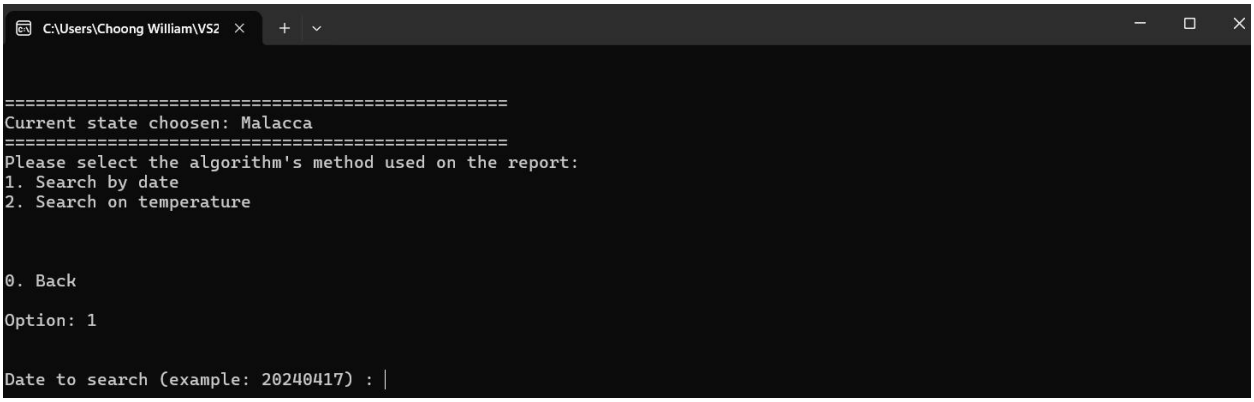

C:\Users\Choong William\VS2
==========
Current state choosen: Malacca
==========
Please select the algorithm's method used on the report:
1. Sort by date in ascending order
2. Sort by date in descending order
3. Sort by highest temperature in ascending order
4. Sort by highest temeperature in descending order
5. Sort by lowest temperature in ascending order
6. Sort by lowest temperature in descending order
0. Back
Option: 3

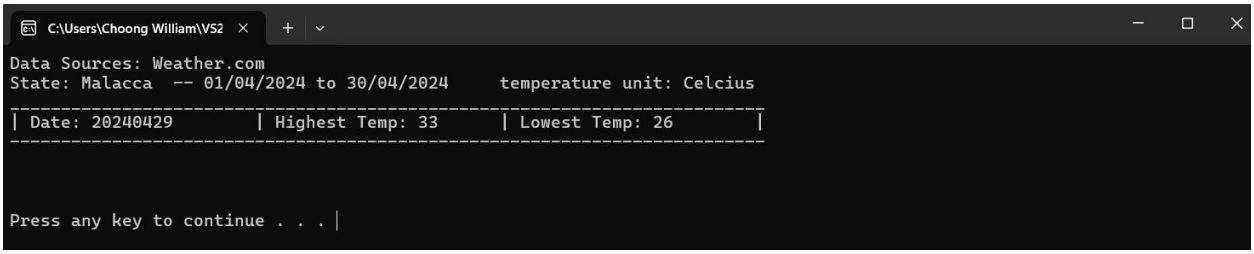

C:\Users\Choong William\VS2
Data Sources: Weather.com
State: Malacca -- 01/04/2024 to 30/04/2024 temperature unit: Celcius
Date: 20240417	Highest Temp: 31	Lowest Temp: 24
Date: 20240408	Highest Temp: 32	Lowest Temp: 26
Date: 20240410	Highest Temp: 32	Lowest Temp: 26
Date: 20240412	Highest Temp: 32	Lowest Temp: 26
Date: 20240420	Highest Temp: 32	Lowest Temp: 27
Date: 20240421	Highest Temp: 32	Lowest Temp: 27
Date: 20240424	Highest Temp: 32	Lowest Temp: 26
Date: 20240428	Highest Temp: 32	Lowest Temp: 26
Date: 20240403	Highest Temp: 33	Lowest Temp: 27
Date: 20240411	Highest Temp: 33	Lowest Temp: 25
Date: 20240416	Highest Temp: 33	Lowest Temp: 24
Date: 20240418	Highest Temp: 33	Lowest Temp: 26
Date: 20240419	Highest Temp: 33	Lowest Temp: 26
Date: 20240422	Highest Temp: 33	Lowest Temp: 26
Date: 20240423	Highest Temp: 33	Lowest Temp: 26
Date: 20240425	Highest Temp: 33	Lowest Temp: 25
Date: 20240426	Highest Temp: 33	Lowest Temp: 26
Date: 20240427	Highest Temp: 33	Lowest Temp: 26
Date: 20240429	Highest Temp: 33	Lowest Temp: 26
Date: 20240401	Highest Temp: 34	Lowest Temp: 26
Date: 20240402	Highest Temp: 34	Lowest Temp: 26
Date: 20240404	Highest Temp: 34	Lowest Temp: 27
Date: 20240405	Highest Temp: 34	Lowest Temp: 27
Date: 20240406	Highest Temp: 34	Lowest Temp: 27
Date: 20240407	Highest Temp: 34	Lowest Temp: 26
Date: 20240409	Highest Temp: 34	Lowest Temp: 26
Date: 20240414	Highest Temp: 34	Lowest Temp: 26
Date: 20240415	Highest Temp: 34	Lowest Temp: 26
Date: 20240430	Highest Temp: 34	Lowest Temp: 26
Date: 20240413	Highest Temp: 35	Lowest Temp: 26
Press any key to continue . . .

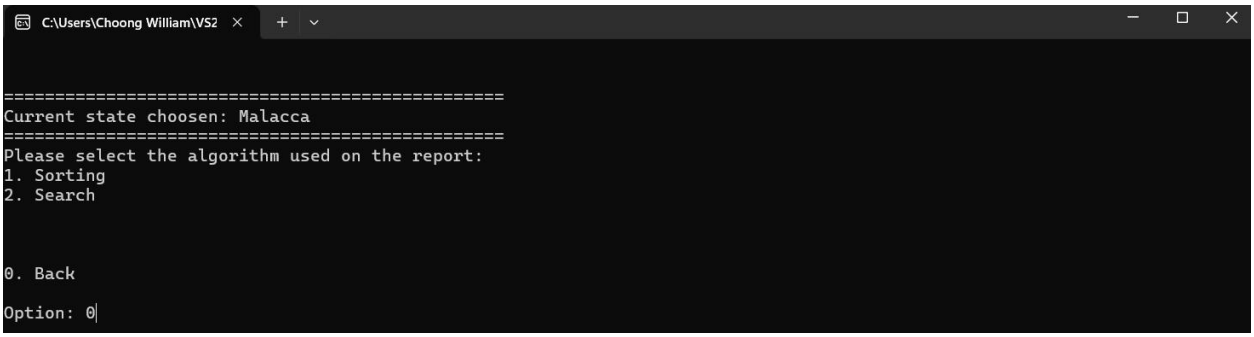

C:\Users\Choong William\VS2
==========
Current state choosen: Malacca
==========
Please select the algorithm's method used on the report:
1. Search by date
2. Search on temperature
0. Back
Option:

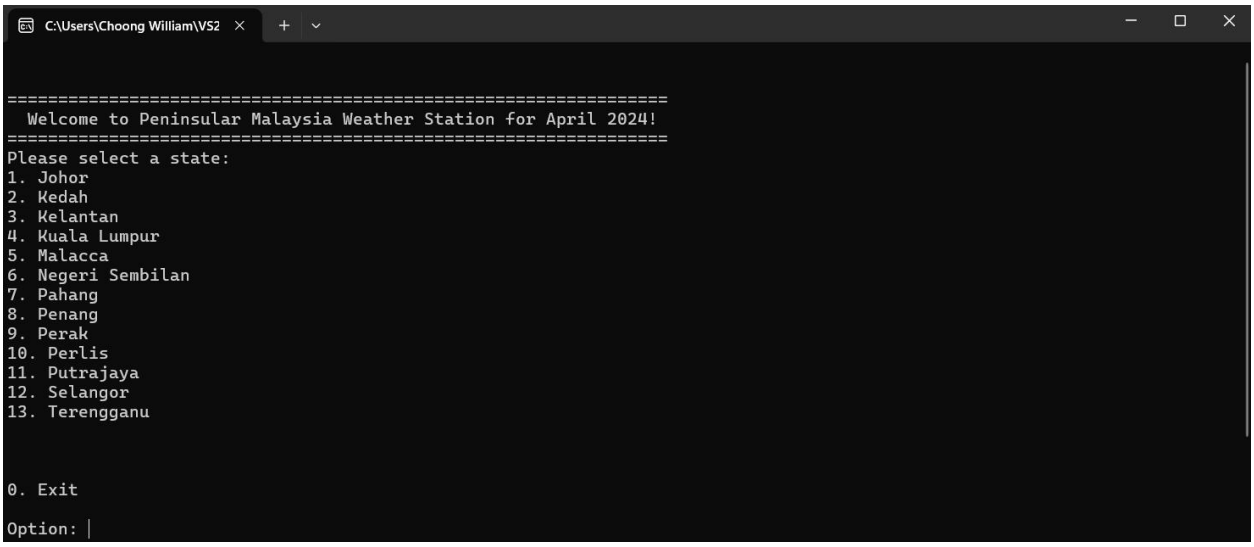

C:\Users\Choong William\VS2
==========
Current state choosen: Malacca
==========
Please select the algorithm used on the report:
1. Sorting
2. Search
0. Back
Option: 2

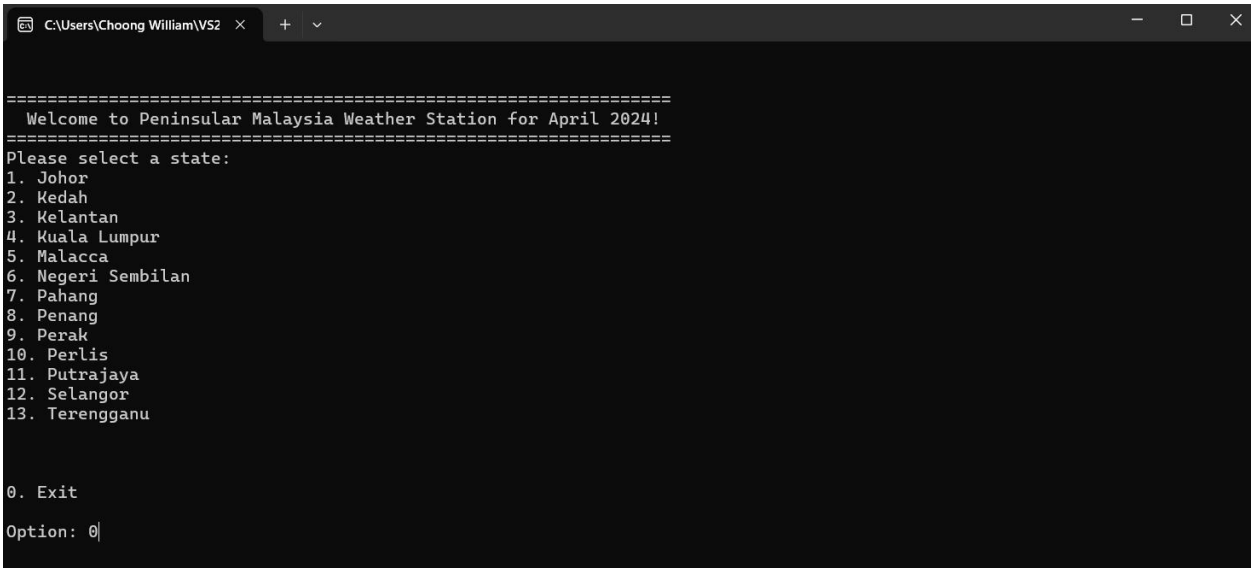

C:\Users\Choong William\VS2
==========
Current state choosen: Malacca
==========
Please select the algorithm's method used on the report:
1. Search by date
2. Search on temperature
0. Back
Option: 1


C:\Users\Choong William\VS2
==========
Current state choosen: Malacca
==========
Please select the algorithm's method used on the report:
1. Search by date
2. Search on temperature
0. Back
Option: 1
Date to search (example: 20240417) :


C:\Users\Choong William\VS2
Data Sources: Weather.com
State: Malacca -- 01/04/2024 to 30/04/2024 temperature unit: Celcius
| Date: 20240429 | Highest Temp: 33 | Lowest Temp: 26 |
Press any key to continue . . .


C:\Users\Choong William\VS2
Current state choosen: Malacca
Please select the algorithm used on the report:
1. Sorting
2. Search
0. Back
Option: 0


C:\Users\Choong William\VS2
Welcome to Peninsular Malaysia Weather Station for April 2024!
Please select a state:
1. Johor
2. Kedah
3. Kelantan
4. Kuala Lumpur
5. Malacca
6. Negeri Sembilan
7. Pahang
8. Penang
9. Perak
10. Perlis
11. Putrajaya
12. Selangor
13. Terengganu
0. Exit
Option:


C:\Users\Choong William\VS2
==========
Welcome to Peninsular Malaysia Weather Station for April 2024!
==========
Please select a state:
1. Johor
2. Kedah
3. Kelantan
4. Kuala Lumpur
5. Malacca
6. Negeri Sembilan
7. Pahang
8. Penang
9. Perak
10. Perlis
11. Putrajaya
12. Selangor
13. Terengganu
0. Exit
Option: 0


Microsoft Visual Studio Debu
Thank you for using our service!
C:\Users\Choong William\VS2022\VS2022\DSA Project\DSA Project\x64\Debug\DSA Project.exe (process 18228) exited with code 0.
Press any key to close this window . . .