동국대학교
dongguk university

제출일	2023.04.04	학과	컴퓨터공학전공
과목	컴퓨터보안	학번	2018112007
담당교수	김영부 교수님	이름	이승현

1) 실습 환경

(1)
운영 체제: Microsoft Windows 11 Home 64bit
프로세서 : Intel(R) Core(TM) i7-10510U @ 1.80GHz (8 CPUs), ~ 2.3GHz
메모리 : DDR4 16GB 2,667MHz
그래픽 카드 : Intel UHD Graphics

(2)
운영 체제: Microsoft Windows 10 Home 64bit
프로세서 : Intel(R) Core(TM) i7-7700HQ @ 2.80GHz (8 CPUs), ~ 2.8GHz
메모리 : DDR4 8GB 2,133MHz
그래픽 카드 : Intel HD Graphics 630, NVIDIA GeForce GTX 1050

2) 실습 진행

1. 문제 분석

(1) 시저 암호

- 시저 암호는 카이사르 암호라고도 하며, 로마 시대의 정치가이자 장군이었던 줄리어스 시저가 처음 사용한 것으로 알려진 간단한 치환 암호의 일종이다.
- 줄리어스 시저는 각 알파벳 순으로 3칸 뒤로 물려 읽는 방법으로 암호문을 작성하였다.

    암호문 : RUSQHUVKBVEHQIIQIYDQJEH
    복호화된 평문 : BECAREFULFORASSASINATOR

- 이번 실습에서는 키워드 7 ~ 10개로 구성된 문장을 생성하고, 1~25 사이의 정수 중 하나를 무작위로 선택하여 그 수만큼 shift를 진행하고 암호문을 생성한다(암호화). 이때 소문자와 대문자를 구분하여 소문자는 소문자 내에서 shift, 대문자는 대문자 내에서 shift를 진행한다.
- 생성된 암호문을 대상으로 암호를 해독하여 평문으로 복호화할 수 있는 일종의 해독기 기능을 구현한다. 이때, shift를 통해 말이 되는 단어와 문장의 후보군을 찾을 수 있어야 한다.
- 이 과정을 100회 반복 후 암호문을 복호화하는데 걸리는 평균 시간을 측정하여 성능 분석을 진행한다.
- 상용화된 시저 암호 해독기 툴이 있다면 이 툴과 내가 구현한 해독기의 성능을 비교 분석한다.

(2) 전치 암호

- 전치 암호는 평문에 나타난 문자 또는 숫자의 기호만 바꾸는 방법으로 평문 문자의 순서를 어떤 특별한 절차에 따라 재배치하고 평문을 암호화하는 방식으로, 순열 암호라고도 한다.
- 대표적으로 단순 전치 암호와 Nihilist 암호가 있다.
- 단순 전치 암호의 경우 평문의 작업 단위를 결정하고, 작업 단위를 맞추기 위해 마지막에 단어를 추가해 평문을 정해진 열 단위로 표현한다. 그 후 자리바꿈 맵을 이용해 행의 순서를 변경한 후 변경된 문자를 열 단위로 표현하여 암호화를 진행한다. 복호화는 이 과정을 반대로 진행하게 된다.
- 이번 실습에서는 시저 암호와 마찬가지로 키워드 7 ~ 10개로 구성된 문장을 생성하고, 생성된 문장에 전치 암호를 적용하여 암호문을 생성한다(암호화).
- 생성된 암호문을 대상으로 암호를 해독하여 평문으로 복호화할 수 있는 일종의 해독기 기능을 구현한다.
- 이 과정을 100회 반복 후 암호문을 복호화하는데 걸리는 평균 시간을 측정하여 성능 분석을 진행한다.
- 상용화된 전치 암호 해독기 툴이 있다면 이 툴과 내가 구현한 해독기의 성능을 비교 분석한다.

2. 프로그램 설계 / 알고리즘

- 시저 암호

(1) 문장 생성 : list로 미리 문장들을 저장한 다음 띄어쓰기를 구분자로 하여 단어들로 분해하고, 벡터에 저장해서 단어 셋을 생성한다. 이때 단어 셋에 중복되는 단어가 없도록 벡터의 내장함수인 unique와 erase를 이용해 중복을 제거하였다. 그 후 단어 셋에서 7~10개의 단어를 선택한 다음에 문장으로 구성한다. 이때 선택하는 단어의 개수는 random 헤더를 include 해서 uniform_int_distribution을 이용해 구현한 random 함수를 이용해 결정한다.

(2) 암호화 : 앞서 구현한 random 함수를 이용해 shift 할 횟수를 결정한 다음 for 문을 이용해 문장들의 문자를 순회하면서 shift를 진행한다. 이때, 소문자와 대문자를 구분하여 소문자는 소문자끼리, 대문자는 대문자끼리 shift를 진행한다. 소문자와 대문자는 islower, isupper를 사용해 구분하였으며, shift를 진행할 때 소문자와 대문자의 범위 밖으로 벗어나지 않도록 % 연산자를 이용해 그 범위 안에서만 shift를 진행되도록 하였다.

(3) 해독 : for 문으로 shift 횟수를 1부터 25까지 조정하면서 문장의 각 문자에 대해 shift를 진행한다. 이때, 문장을 구성하는 각 단어가 단어 셋에 존재하는지 확인하여 5개 이상이 단어 셋에 존재하는 경우 출력하는 식으로 후보군을 추려낸다. 또한, 해독되었는데 또 해독을 시도하는 경우 해독이 완료되었다는 메시지를 출력한다.

- 전치 암호

(1) 문장 생성 : list로 미리 문장들을 저장한 다음 띄어쓰기를 구분자로 하여 단어들로 분해하고, 벡터에 저장해서 단어 셋을 생성한다. 이때 단어 셋에 중복되는 단어가 없도록 벡터의 내장함수인 unique와 erase를 이용해 중복을 제거하였다. 그 후 단어 셋에서 7~10개의 단어를 선택한 다음에 문장으로 구성한다. 이때 선택하는 단어의 개수는 random 헤더를 include 해서 uniform_int_distribution을 이용해 구현한 random 함수를 이용해 결정한다.

(2) 암호화 : 생성한 문장들의 문자를 순회하는데, % 연산자를 이용해 행 순서로 읽고 그룹별로 나누어 저장한다. 이때, 5개의 그룹으로 나눌 것이기 때문에 index % 5를 하여 5개 단위로 문자를 읽는다. 그 후 pair<int, string>를 사용해 그룹별로 나누어진 문장들의 순서와 문장들을 저장하는 맵을 구성하고, random_shuffle 함수로 무작위로 섞은 다음 열 단위로 표현한다.

(3) 해독 : 암호화할 때 5개의 그룹으로 나누었기 때문에, 해독할 때도 암호문을 다시 열 단위 읽어, 5개의 그룹으로 분리한 다음, 앞서 저장된 맵과 분리된 문장을 대조하여 다시 제 순서대로 정렬한다. 그 후 정렬된 문장들을 행 순서로 읽으면서 평문을 복구한다. 또한, 해독되었는데 또 해독을 시도하는 경우 해독이 완료되었다는 메시지를 출력한다.

- GUI

winform을 이용하여 간단하게 GUI를 구현해보았다. winform은 마이크로소프트의 닷넷 프레임워크에서 제공하며, visual studio에서 GUI를 구성할 수 있다. C++과 C#을 이용해 GUI의 기능을 설정할 수 있으며, 나는 C++을 이용해 GUI를 구성하였다. 이때 CLI를 따로 설치해야 하므로 C#보다 번거로움은 있었지만, C#에 익숙지 않다면 CLI를 통해서 C++로도 구현할 수 있기에 이를 위해서라면 어쩔 수 없다. GUI를 구성하는 요소들은 왼쪽의 도구 상자에서 끌어다가 원하는 위치에 배치할 수 있으며, 요소별로 이벤트를 설정해줄 수 있다. 이 이벤트에는 자기가 작성한 함수도 포함하기 때문에 원하는 기능을 구현하여 요소에 지정할 수 있다. 나는 버튼을 배치한 다음, 문장 생성과 암호화 기능, 해독 기능, 이를 100회 반복하는 기능을 버튼별로 부여하여 버튼을 클릭할 때마다 실행될 수 있도록 하였다.

3. 소스 코드 / 주석

(1) 시저 암호

a) code.h

```
#include <iostream>
#include <string>
#include <vector>
#include <random>
#include <list>
#include <ctime>
#include <sstream>
#pragma once
using namespace std;
using namespace System;
using namespace System::Windows::Forms;
class CaeserCiper
{
		private:
			string sentense, origin; //원 문장과 암호화 되는 문장을 저장
			vector<string> words;		//단어셋 저장
			bool decrypted =false;		//해독이 되었는지 확인하는 플래그
			list<string> dictionory = { "THE CLOCK STOPPED TICKING FOREVER AGO",
				"The trembling fear",
				"I have a story to tell",
				"Do you hear me tonight",
				"I feel you drop tears",
				"And my heart become heavy"
				"Some lights turn around and around",
				"I lost my head again",
			"And just want to throw everything away",
			"Please tell me what should I do",
			"You made me feel so better"}; //단어셋을 구성하기 위한 문장 구성
			vector<double> sum_time;	//해독하는데 소모하는 시간 저장
		private: String^ ToGcString(string std_string) {
			return gcnew String(std_string.data()); //std::string 에서 System::String으로 변환 }
								//winform에서 std::string을 출력할 수 없어서 System::String으로 변환 필요
		private: string ToStdString(String^ gc_string) {
			using namespace Runtime::InteropServices;
			const char* chars = (const char*)(Marshal::StringToHGlobalAnsi(gc_string)).ToPointer();
			string std_string = chars;
			Marshal::FreeHGlobal(IntPtr((void*)chars));
			return std_string; //System::String을 std::string으로 변환
		}
		public:
			CaeserCiper()
			{
				for (auto l : dictionory)
				{
					vector<string> temp = split(l, ' '); //띄어쓰기를 delimeter로 하여 단어로 분해
					words.insert(words.end(), temp.begin(), temp.end()); //단어 저장
				}
				sort(words.begin(), words.end()); //단어셋 정렬
				words.erase(unique(words.begin(), words.end()), words.end()); //중복 제거
			}
			int random(int low, int high)
			{
				random_device rd;
				mt19937 gen(rd());
				uniform_int_distribution<int> digit(low, high); //정수를 균일한 확률로 무작위로 선택
				int ran = digit(gen);		//무작위로 선택된 정수를 저장
				return ran;
			}
			void createSentense()
			{
				decrypted =false;
				int index = random(7, 10); //문장을 구성할 단어의 개수 정하기
				string temp ="";
				for (int i =0; i < index; i++)
				{
					temp += words[random(0, words.size() -1)];	//단어를 무작위로 선택해 저장
					temp +=' '; //띄어쓰기로 구분
				}
				sentense = temp; //생성된 문장 저장
				origin = sentense; //암호화 전 문장 저장
			}
			vector<string> split(string str, char delimeter)
			{
				vector<string> vec;
				stringstream ss(str);
				string s;
				while (getline(ss, s, delimeter)) vec.push_back(s);
				//문장을 입력 받아 delimeter를 기준으로 분리하고 저장
				return vec;
			}
			void encrypt()
			{
				int shift = random(1, 25); //shift 횟수 구하기
				for (auto &s : sentense)
				{
					if (islower(s)) s = (((s -97) + shift) % 26) +97; //소문자 일 경우 소문자끼리 shift
					if (isupper(s)) s = (((s -65) + shift) % 26) +65; //대문자 일 경우 대문자끼리 shift
				}
			}
			void decrypt(TextBox^ Box)
			{
				for (int shift =1; shift <26; shift++) //반복문으로 shift 횟수 조절
				{
					for (auto &s : sentense)
					{
						if (islower(s)) s = (((s -97) +1) % 26) +97; //소문자 일 경우 소문자끼리 shift하여 해독
						if (isupper(s)) s = (((s -65) +1) % 26) +65; //대문자 일 경우 대문자끼리 shift하여 해독
					}
					if (search(sentense) >5) Box -> AppendText(ToGcString(sentense) + Environment::NewLine + Environment::NewLine);
					//문장을 구성하는 단어 중 5개 이상이 사전에 만든 단어셋에 존재할 경우 출력
				}
				decrypted =true; //해독이 완료되었음을 플래그로 표현
			}
			void run(TextBox^ Box) //암호화와 해독을 100회 반복하기 위해 정의
			{
				sum_time.clear(); //전에 저장한 시간 정보들 삭제
				clock_t start, finish; //시작 시간과 끝나는 시간을 저장하기 위한 변수
				double duration; //소요시간을 저장하기 위한 변수
				for (int i =0; i <100; i++) {
					start = clock(); //시작 시간 저장
					createSentense(); //문장 생성
					Box->Text ="<Created Random Sentense!>"+ Environment::NewLine;
					Box->AppendText(getSentense() + Environment::NewLine + Environment::NewLine);
					//생성된 문장 출력
					encrypt(); //암호화 실행
					Box->AppendText("<Encrypt Caeser Ciper!>"+ Environment::NewLine);
					Box->AppendText(getSentense() + Environment::NewLine + Environment::NewLine);
					//암호화된 문장 출력
```

```
                                        Box->AppendText("<Decrypt Caeser Ciper!>"+ Environment::NewLine);
                                        decrypt(Box); //해독 실행
                                        finish = clock(); //끝나는 시간 저장
                                        duration = (double)(finish - start); //소요시간 저장
                                        sum_time.push_back(duration); //소요시간을 벡터에 저장
                                }
                                double sum =0; //시간의 합을 저장하기 위한 변수
                                for (auto a : sum_time) sum += a; //소요시간 더하기
                                Box->AppendText("Caeser Ciper 평균 시간 : "+ ((sum / sum_time.size()) / CLOCKS_PER_SEC)+" sec"+ Environment::NewLine);
                                //100회 해독하는데 소요된 평균 시간 출력
                        }
                        int search(string sentense)
                        {
                                int count =0;
                                vector<string> vec = split(sentense, ' '); //문장을 delimeter 기준으로 분해
                                for (auto a : words) { //단어셋 순회
                                        if (find(vec.begin(), vec.end(), a) != vec.end()) count++;
                                        //구성된 문장의 단어가 단어셋에 존재하는지 확인
                                }
                                return count; //단어셋에 존재하는 단어의 개수 반환
                        }
                        String^ getSentense()
                        {
                                return ToGcString(sentense); //암호화 과정에서의 문장 출력
                        }
                        bool check_decrypted()
                        {
                                return decrypted; //해독 여부 플래그값 반환
                        }
};
```

(b) GUI.h

```
#pragma once
#include "code.h"
using namespace std;
namespace Caeser {
        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        CaeserCiper caeser;
        /// <summary>
        /// GUI에 대한 요약입니다.
        /// </summary>
        public ref class GUI : public System::Windows::Forms::Form
        {
        public:
                GUI(void)
                {
                        InitializeComponent();
                        //
                        //TODO: 생성자 코드를 여기에 추가합니다.
                        //
                        CheckForIllegalCrossThreadCalls =false;
                }
        protected:
                /// <summary>
                /// 사용 중인 모든 리소스를 정리합니다.
                /// </summary>
                ~GUI()
                {
                        if (components)
                        {
                                delete components;
                        }
                }
        private: System::Windows::Forms::Button^ button1;
        private: System::Windows::Forms::Button^ button2;
        private: System::Windows::Forms::TextBox^ textBox1;
        private: System::Windows::Forms::Button^ button3;
        protected:
        private:
                /// <summary>
                /// 필수 디자이너 변수입니다.
                /// </summary>
                System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
                /// <summary>
                /// 디자이너 지원에 필요한 메서드입니다.
                /// 이 메서드의 내용을 코드 편집기로 수정하지 마세요.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->button1 = (gcnew System::Windows::Forms::Button());
                        this->button2 = (gcnew System::Windows::Forms::Button());
                        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                        this->button3 = (gcnew System::Windows::Forms::Button());
                        this->SuspendLayout();
                        //
                        // button1
                        //
                        this->button1->Location =System::Drawing::Point(71, 50);
                        this->button1->Name = L"button1";
                        this->button1->Size =System::Drawing::Size(125, 89);
                        this->button1->TabIndex =0;
                        this->button1->Text = L"Caeser Encrypt";
                        this->button1->UseVisualStyleBackColor =true;
                        this->button1->Click += gcnew System::EventHandler(this, &GUI::button1_Click);
                        //
                        // button2
                        //
                        this->button2->Location =System::Drawing::Point(256, 50);
                        this->button2->Name = L"button2";
                        this->button2->Size =System::Drawing::Size(125, 89);
                        this->button2->TabIndex =1;
                        this->button2->Text = L"Caeser Decrypt";
                        this->button2->UseVisualStyleBackColor =true;
                        this->button2->Click += gcnew System::EventHandler(this, &GUI::button2_Click);
                        //
                        // textBox1
                        //
                        this->textBox1->Location =System::Drawing::Point(71, 189);
                        this->textBox1->Multiline =true;
                        this->textBox1->Name = L"textBox1";
                        this->textBox1->ReadOnly =true;
                        this->textBox1->ScrollBars =System::Windows::Forms::ScrollBars::Vertical;
                        this->textBox1->Size =System::Drawing::Size(488, 234);
                        this->textBox1->TabIndex =2;
                        this->textBox1->TextAlign =System::Windows::Forms::HorizontalAlignment::Center;
                        //
```

```
                                        // button3
                                        //
                                        this->button3->Location = System::Drawing::Point(434, 50);
                                        this->button3->Name = L"button3";
                                        this->button3->Size = System::Drawing::Size(125, 89);
                                        this->button3->TabIndex = 3;
                                        this->button3->Text = L"Automatic Run";
                                        this->button3->UseVisualStyleBackColor = true;
                                        this->button3->Click += gcnew System::EventHandler(this, &GUI::button3_Click);
                                        //
                                        // GUI
                                        //
                                        this->AutoScaleDimensions = System::Drawing::SizeF(7, 12);
                                        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                                        this->ClientSize = System::Drawing::Size(640, 435);
                                        this->Controls->Add(this->button3);
                                        this->Controls->Add(this->textBox1);
                                        this->Controls->Add(this->button2);
                                        this->Controls->Add(this->button1);
                                        this->Name = L"GUI";
                                        this->Text = L"Caeser Ciper";
                                        this->ResumeLayout(false);
                                        this->PerformLayout();
                                }
#pragma endregion
                private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
                                caeser.createSentense(); //문장 생성
                                textBox1->Text ="<Created Random Sentense!>"+ Environment::NewLine;
                                textBox1->AppendText(caeser.getSentense() + Environment::NewLine + Environment::NewLine);
                                //생성된 문장 출력
                                caeser.encrypt(); //생성된 문장을 암호화
                                textBox1->AppendText("<Encrypt Caeser Ciper!>"+ Environment::NewLine);
                                textBox1->AppendText(caeser.getSentense() + Environment::NewLine + Environment::NewLine);
                                //암호화된 문장 출력
                }
                private: void run()
                {
                                caeser.decrypt(textBox1); //암호화된 문장 해독
                }
                private: void all_run()
                {
                                caeser.run(textBox1); //암호화와 해독 100회 반복
                }
                private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
                                if (!caeser.check_decrypted()) //해독되지 않았다면
                                {
                                                textBox1->AppendText("<Decrypt Caeser Ciper!>"+ Environment::NewLine);
                                                System::Threading::Tasks::Task::Factory->StartNew(gcnew Action(this, &GUI::run));
                                                //스레드를 사용해 해독 실행, 병렬 처리를 통해 빠른 해독, 끊김없는 GUI 출력
                                }
                                else
                                                textBox1->AppendText("<Decrypted!>"+ Environment::NewLine); //해독되었음을 출력
                }
                private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
                                System::Threading::Tasks::Task::Factory->StartNew(gcnew Action(this, &GUI::all_run));
                                //100회 암호화와 해독 반복, 스레드를 사용해 해독 실행, 병렬 처리를 통해 빠른 해독, 끊김없는 GUI 출력
                }
};
}
```

(c) GUI.cpp

```
#include "GUI.h"
using namespace System;
using namespace System::Windows::Forms;
[STAThreadAttribute]
void main() {
                Application::SetCompatibleTextRenderingDefault(false);
                Application::EnableVisualStyles();
                Caeser::GUI form;                       // 프로젝트내 GUI 파일 선언
                Application::Run(% form); //GUI 실행
}
```

(2) 전치 암호

a) code.h

```
#include <iostream>
#include <string>
#include <vector>
#include <random>
#include <list>
#include <map>
#include <algorithm>
#include <ctime>
#include <sstream>
#pragma once
using namespace std;
using namespace System;
using namespace System::Windows::Forms;
class TranspositionCiper
{
private:
                bool decrypted =false; //해독이 되었는지 확인하는 플래그
                int mismatch =0;
                string sentense, origin_sentense; //원 문장과 암호화 되는 문장을 저장
                vector<string> words; //단어셋 저장
                list<string> dictionory = { "THE CLOCK STOPPED TICKING FOREVER AGO",
                                                                                                "The trembling fear",
                                                                                                "I have a story to tell",
                                                                                                "Do you hear me tonight",
                                                                                                "I feel you drop tears",
                                                                                                "And my heart become heavy"
                                                                                                "Some lights turn around and around",
                                                                                                "I lost my head again",
                                                                                                "And just want to throw everything away",
                                                                                                "Please tell me what should I do",
                                                                                                "You made me feel so better" };

                vector<pair<int, string>> change_map; //자리바꿈맵
                vector<double> sum_time; //해독하는데 소모하는 시간 저장
private: String^ ToGcString(string std_string) {
                return gcnew String(std_string.data()); //std::string 에서 System::String으로 변환
}                                                                                                                                                                                                                                               //winform에서  std::string을
출력할 수 없어서 System::String으로 변환 필요
private: string ToStdString(String^ gc_string) {
                using namespace Runtime::InteropServices;
                const char* chars = (const char*)(Marshal::StringToHGlobalAnsi(gc_string)).ToPointer();
                string std_string = chars;
                Marshal::FreeHGlobal(IntPtr((void*)chars));
                return std_string; //System::String을 std::string으로 변환
```

```
}
public:
		TranspositionCiper()
		{
			for (auto l : dictionory)
			{
				vector<string> temp = split(l, ' '); //띄어쓰기를 delimeter로 하여 단어로 분해
				words.insert(words.end(), temp.begin(), temp.end()); //단어 저장
			}
			sort(words.begin(), words.end()); //단어셋 정렬
			words.erase(unique(words.begin(), words.end()), words.end()); //중복 제거
		}
		int random(int low, int high)
		{
			random_device rd;
			mt19937 gen(rd());
			uniform_int_distribution<int> digit(low, high); //정수를 균일한 확률로 무작위로 선택
			int ran = digit(gen);                           //무작위로 선택된 정수를 저장
			return ran;
		}
		void createSentense()
		{
			decrypted =false;
			int index = random(7, 10); //문장을 구성할 단어의 개수 정하기
			string temp ="";
			for (int i =0; i < index; i++)
			{
				temp += words[random(0, words.size() -1)]; //단어를 무작위로 선택해 저장
				temp +=' '; //띄어쓰기로 구분
			}
			sentense = temp; //생성된 문장 저장
			change_map.clear();
			if (sentense.length() % 5 !=0) //문장을 5개 그룹으로 나누는데 마지막 그룹에서 빈칸이 있는 경우
			{
				for (; sentense.length() % 5 !=0;)
					sentense += (random(0, 1) ? 65 + random(0, 25) : 97 + random(0, 25));
					//소문자와 대문자를 섞어서 뒤에 붙임
			}
			origin_sentense = sentense; //알파벳 붙인 문장 저장
		}
		vector<string> split(string str, char delimeter)
		{
			vector<string> vec;
			stringstream ss(str);
			string s;
			while (getline(ss, s, delimeter)) vec.push_back(s);
			//문장을 입력 받아 delimeter를 기준으로 분리하고 저장
			return vec;
		}
		void encrypt()
		{
			vector<string> array = { "", "", "", "", "" };
			//다섯 그룹으로 나뉜 문장을 저장할 벡터
			for (int i =0; i < sentense.length(); i++)
				array[i % 5] += sentense[i];
				//행 순서로 읽기
			for (int i =0; i < array.size(); i++)
				change_map.push_back(pair<int, string> (i, array[i]));
				//다섯 그룹으로 나뉜 문장들을 저장
			random_shuffle(change_map.begin(), change_map.end());
			//무작위로 섞음
			sentense.clear();
			//문장 초기화
			for (auto a : change_map) sentense += a.second;
			//섞인 문장들을 이어붙여 암호화
		}
		void decrypt(TextBox^ Box)
		{
			vector<string> array;
			for (int i =0; i < sentense.length(); i += (sentense.length() /5))
				array.push_back(sentense.substr(i, (sentense.length() /5)));
				//암호화된 문장들을 다시 다섯 그룹으로 분해
			vector<string> origin = { "", "", "", "", "" };
			//분해된 문장들 저장
			for (int i =0; i < array.size(); i++)
			{
				for (auto p : change_map)
				{
					if (p.second == array[i]) origin[p.first] = array[i];
					//맵에 저장된 값에 따라 분해된 문장들 다시 정렬
				}
			}
			sentense.clear();
			for (int i =0; i < origin[0].length(); i++)
			{
				for (int j =0; j < origin.size(); j++)
				{
					sentense += origin[j][i];
					//정렬된 문장들을 다시 행순서로 읽고 복원
				}
			}
			if (origin_sentense == sentense)
				Box->AppendText(ToGcString(sentense) + Environment::NewLine + Environment::NewLine);
				//해독이 잘 되었다면 해독된 문장 출력
			else mismatch++; //해독 실패 횟수 증가
			decrypted =true; //해독이 완료되었음을 플래그로 표현
		}
		void run(TextBox^ box) //암호화와 해독을 100회 반복하기 위해 정의
		{
			sum_time.clear(); //전에 저장한 시간 정보들 삭제
			clock_t start, finish; //시작 시간과 끝나는 시간을 저장하기 위한 변수
			double duration; //소요시간을 저장하기 위한 변수

			for (int i =0; i <100; i++)
			{
				start = clock(); //시작 시간 저장
				createSentense(); //문장 생성
				box->Text ="<Created Random Sentense!>"+ Environment::NewLine;
				box->AppendText(getSentense() + Environment::NewLine + Environment::NewLine);
				//생성된 문장 출력
				encrypt(); //암호화 실행
				box->AppendText("<Encrypt Transposition Ciper!>"+ Environment::NewLine);
				box->AppendText(getSentense() + Environment::NewLine + Environment::NewLine);
				//암호화된 문장 출력
				box->AppendText("<Decrypt Transposition Ciper!>"+ Environment::NewLine);
				decrypt(box); //해독 실행
				finish = clock(); //끝나는 시간 저장
				duration = (double)(finish - start); //소요시간 저장
				sum_time.push_back(duration);//소요시간을 벡터에 저장
			}
			double sum =0; //시간의 합을 저장하기 위한 변수
			for (auto a : sum_time) sum += a; //소요시간 더하기
			box->AppendText("Transposition Ciper 평균 시간 : "+ ( (sum / sum_time.size()) / CLOCKS_PER_SEC) +" sec"+ Environment::NewLine);
			//100회 해독하는데 소요된 평균 시간 출력
			box->AppendText("mismatch count = "+ mismatch);
			//해독 실패 횟수 출력
```

```
		}
		String^ getSentense()
		{
			return ToGcString(sentense); //암호화 과정에서의 문장 출력
		}
		bool check_decrypted()
		{
			return decrypted; //해독 여부 플래그값 반환
		}
};
```

(b) GUI.h

```
#pragma once
#include "code.h"
namespace Transposition {
		using namespace System;
		using namespace System::ComponentModel;
		using namespace System::Collections;
		using namespace System::Windows::Forms;
		using namespace System::Data;
		using namespace System::Drawing;
		/// <summary>
		/// GUI에 대한 요약입니다.
		/// </summary>

		TranspositionCiper trans;

		public ref class GUI : public System::Windows::Forms::Form
		{
		public:
			GUI(void)
			{
				InitializeComponent();
				//
				//TODO: 생성자 코드를 여기에 추가합니다.
				//
				CheckForIllegalCrossThreadCalls =false;
			}
		protected:
			/// <summary>
			/// 사용 중인 모든 리소스를 정리합니다.
			/// </summary>
			~GUI()
			{
				if (components)
				{
					delete components;
				}
			}
		private: System::Windows::Forms::Button^ button1;
		private: System::Windows::Forms::Button^ button2;
		private: System::Windows::Forms::TextBox^ textBox1;
		private: System::Windows::Forms::Button^ button3;
		protected:
		private:
			/// <summary>
			/// 필수 디자이너 변수입니다.
			/// </summary>
			System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
			/// <summary>
			/// 디자이너 지원에 필요한 메서드입니다.
			/// 이 메서드의 내용을 코드 편집기로 수정하지 마세요.
			/// </summary>
			void InitializeComponent(void)
			{
				this->button1 = (gcnew System::Windows::Forms::Button());
				this->button2 = (gcnew System::Windows::Forms::Button());
				this->textBox1 = (gcnew System::Windows::Forms::TextBox());
				this->button3 = (gcnew System::Windows::Forms::Button());
				this->SuspendLayout();
				//
				// button1
				//
				this->button1->Location =System::Drawing::Point(42, 47);
				this->button1->Name = L"button1";
				this->button1->Size =System::Drawing::Size(143, 97);
				this->button1->TabIndex =1;
				this->button1->Text = L"Transposition Encrypt";
				this->button1->UseVisualStyleBackColor =true;
				this->button1->Click += gcnew System::EventHandler(this, &GUI::button1_Click);
				//
				// button2
				//
				this->button2->Location =System::Drawing::Point(251, 47);
				this->button2->Name = L"button2";
				this->button2->Size =System::Drawing::Size(143, 97);
				this->button2->TabIndex =2;
				this->button2->Text = L"Transposition Decrypt";
				this->button2->UseVisualStyleBackColor =true;
				this->button2->Click += gcnew System::EventHandler(this, &GUI::button2_Click);
				//
				// textBox1
				//
				this->textBox1->Location =System::Drawing::Point(42, 178);
				this->textBox1->Multiline =true;
				this->textBox1->Name = L"textBox1";
				this->textBox1->ReadOnly =true;
				this->textBox1->ScrollBars =System::Windows::Forms::ScrollBars::Vertical;
				this->textBox1->Size =System::Drawing::Size(553, 226);
				this->textBox1->TabIndex =3;
				this->textBox1->TextAlign =System::Windows::Forms::HorizontalAlignment::Center;
				this->textBox1->TextChanged += gcnew System::EventHandler(this, &GUI::textBox1_TextChanged);
				//
				// button3
				//
				this->button3->Location =System::Drawing::Point(452, 47);
				this->button3->Name = L"button3";
				this->button3->Size =System::Drawing::Size(143, 97);
				this->button3->TabIndex =4;
				this->button3->Text = L"Autumatic Run";
				this->button3->UseVisualStyleBackColor =true;
				this->button3->Click += gcnew System::EventHandler(this, &GUI::button3_Click);
				//
				// GUI
				//
				this->AutoScaleDimensions =System::Drawing::SizeF(7, 12);
				this->AutoScaleMode =System::Windows::Forms::AutoScaleMode::Font;
				this->ClientSize =System::Drawing::Size(640, 457);
				this->Controls->Add(this->button3);
				this->Controls->Add(this->textBox1);
				this->Controls->Add(this->button2);
				this->Controls->Add(this->button1);
```

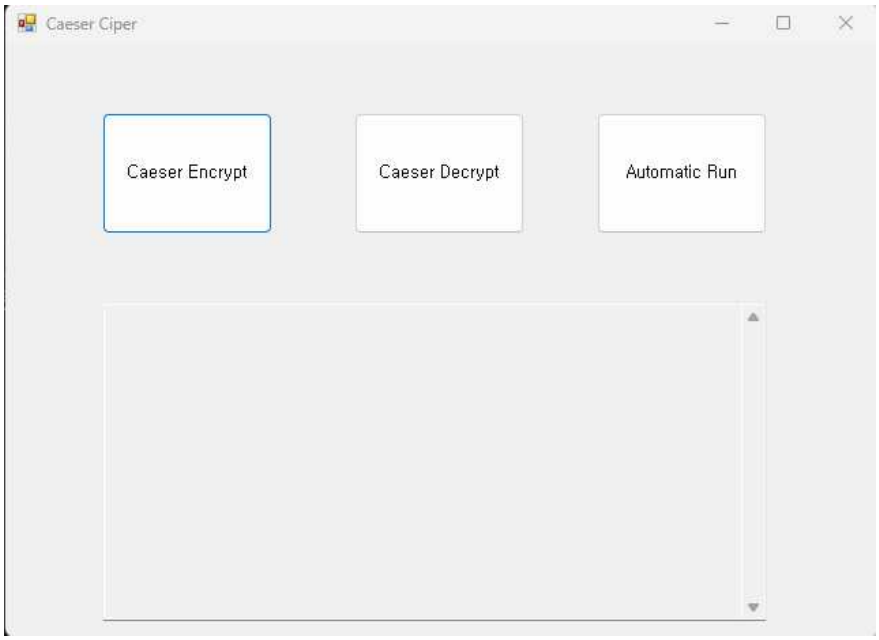
```
                this->Name = L"GUI";
                this->Text = L"Transposition Ciper";
                this->ResumeLayout(false);
                this->PerformLayout();
            }
#pragma endregion
        private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
                trans.createSentense(); //문장 생성
                textBox1->Text ="<Created Random Sentense!>"+ Environment::NewLine;
                textBox1->AppendText(trans.getSentense() + Environment::NewLine + Environment::NewLine);
                //생성된 문장 출력
                trans.encrypt(); //생성된 문장을 암호화
                textBox1->AppendText("<Encrypt Transposition Ciper!>"+ Environment::NewLine);
                textBox1->AppendText(trans.getSentense() + Environment::NewLine + Environment::NewLine);
                //암호화된 문장 출력
        }
        private: void run()
        {
                trans.decrypt(textBox1); //암호화된 문장 해독
        }
        private: void all_run()
        {
                trans.run(textBox1); //암호화와 해독 100회 반복
        }
        private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {

                if (!trans.check_decrypted()) //해독되지 않았다면
                {
                        textBox1->AppendText("<Decrypt Transposition Ciper!>"+ Environment::NewLine);
                        System::Threading::Tasks::Task::Factory->StartNew(gcnew Action(this, &GUI::run));
                        //스레드를 사용해 해독 실행, 병렬 처리를 통해 빠른 해독, 끊김없는 GUI 출력
                }
                else
                        textBox1->AppendText("<Decrypted>"+ Environment::NewLine); //해독되었음을 출력
        }
        private: System::Void textBox1_TextChanged(System::Object^ sender, System::EventArgs^ e) {
        }
        private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
                System::Threading::Tasks::Task::Factory->StartNew(gcnew Action(this, &GUI::all_run));
                //100회 암호화와 해독 반복, 스레드를 사용해 해독 실행, 병렬 처리를 통해 빠른 해독, 끊김없는 GUI 출력
        }
};
}
```

(c) GUI.cpp

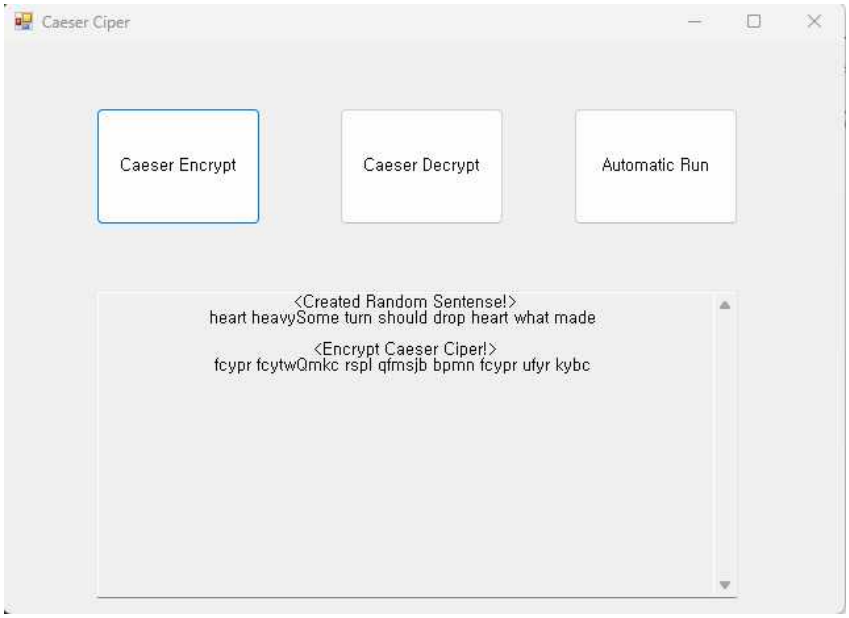
```
#include "GUI.h"
using namespace System;
using namespace System::Windows::Forms;
[STAThreadAttribute]
void main() {
        Application::SetCompatibleTextRenderingDefault(false);
        Application::EnableVisualStyles();
        Transposition::GUI form;        // 프로젝트내 GUI 파일 선언
        Application::Run(% form); //GUI 실행
}
```

4. 결과 / 결과 분석

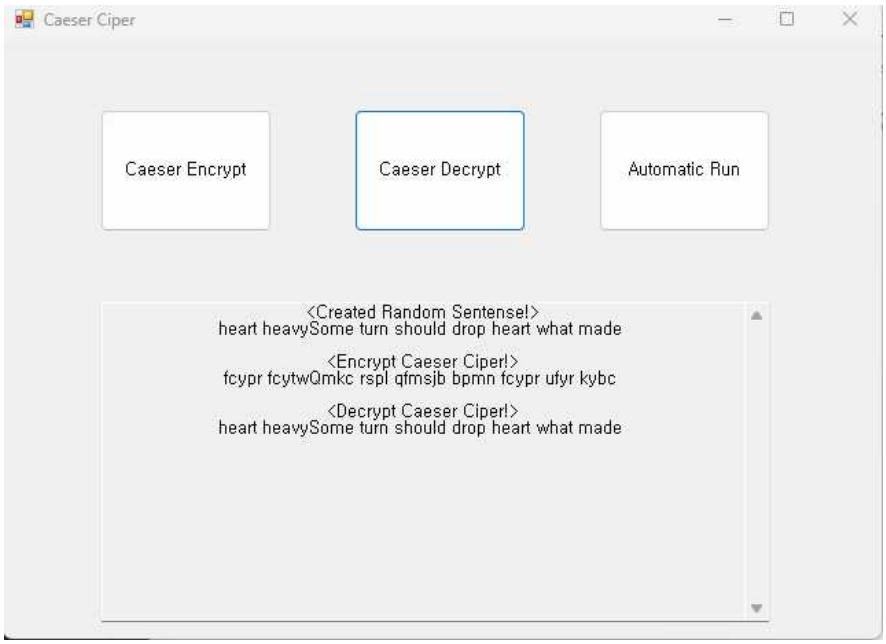
(1) 시저 암호



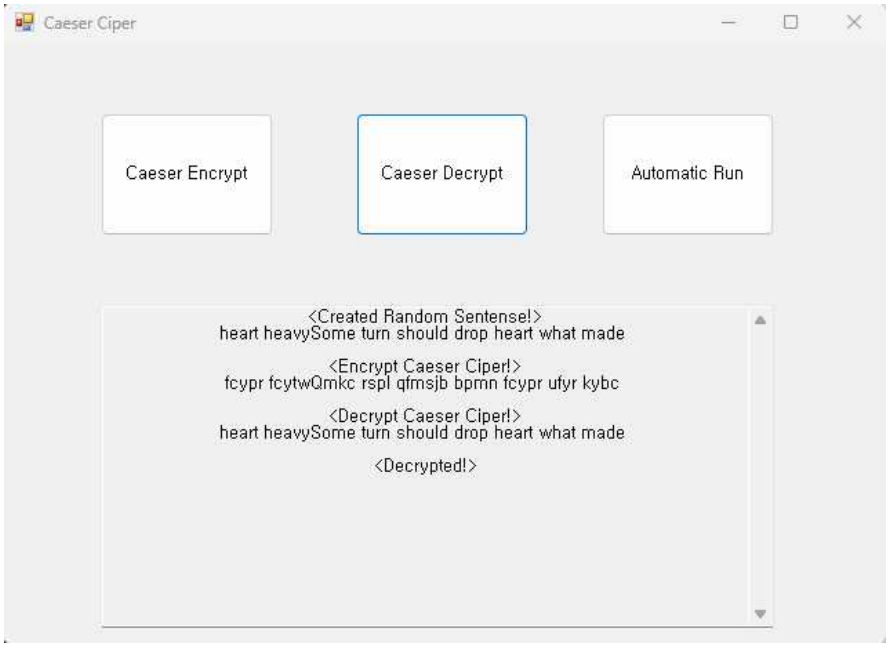
- 시저 암호를 이용해 암호화와 해독을 수행하는 프로그램의 GUI 모습이다.
- Caesar Encrypt 버튼을 클릭하면 문장 하나를 무작위로 생성하고, 생성된 문장의 암호화를 수행한다.
- Caesar Decrypt 버튼을 클릭하면 앞서 생성된 암호문의 해독을 수행한다.
- Automatic Run 버튼을 클릭하면 시저 암호의 암호화와 해독을 100회 반복 수행하고, 평균 시간을 출력한다.



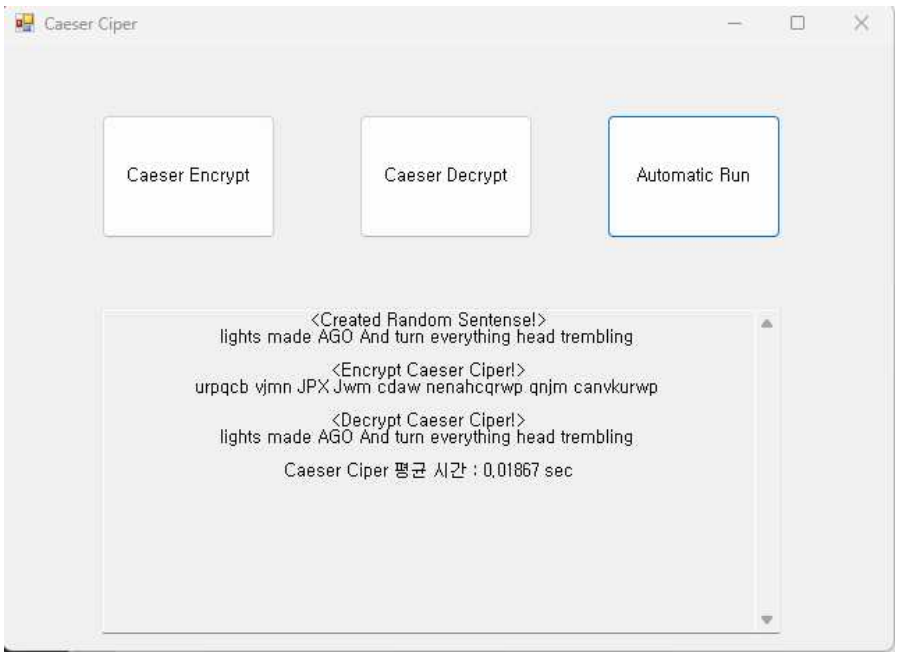
- Caesar Encrypt를 클릭한 모습이다.
- 처음에 heart heavySome turn should drop heart what made라는 문장이 무작위로 생성되었으며, 이 문장을 시저 암호를 이용하여 암호화를 진행하였다.
- 생성된 암호문은 fcypr fcytwQmkc rspl qfmsjb bpmn fcypr ufyr kybc 이다.
- 암호문을 생성하는데 shift 한 수는 무작위로 결정되며 소문자는 소문자끼리, 대문자는 대문자끼리 shift가 진행된 모습을 평문과 암호문을 비교하면 볼 수 있다. (heavySome -> fcytwQmkc)
- 문장 하나만 생성된 모습을 볼 수 있다.



- 앞서 생성된 암호문을 해독한 모습이다.
- 해독이 완료된 평문과 암호화를 하기 전 평문을 비교하면 완벽하게 해독된 모습을 볼 수 있다.
- 해독되는 원리는 1~25번 shift를 진행하면서 각 문장을 구성하는 단어들이 단어 셋에 저장되어있는 단어와 얼마나 유사한지 확인하고, 5개 이상의 단어들이 단어 셋에 저장되어있으면 후보군으로써 출력하게 된다.

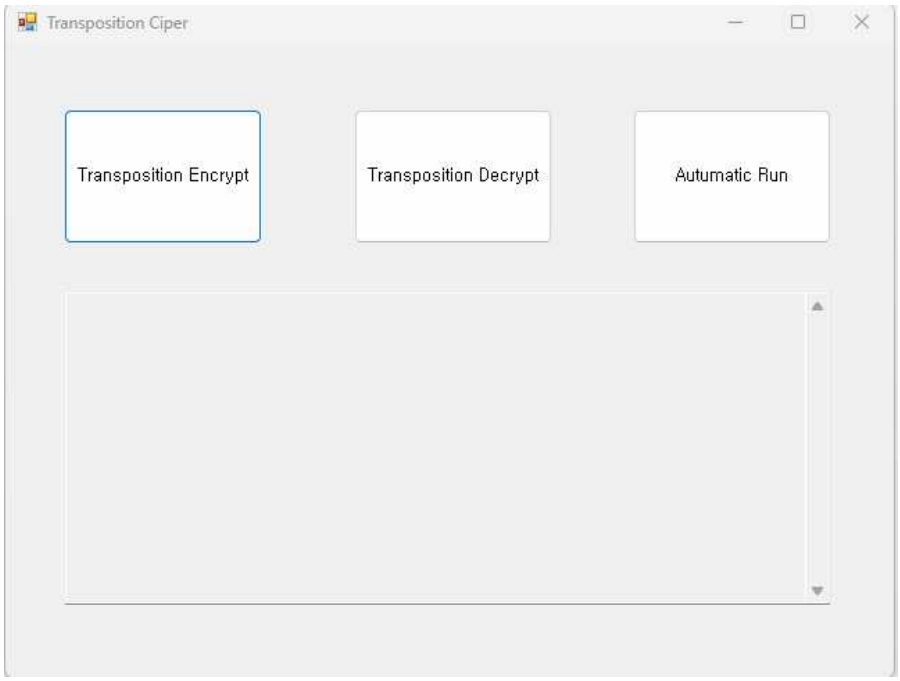


- 해독이 완료된 상태에서 다시 Caesar Decrypt 버튼을 클릭하면 해독이 완료되었음을 알리는 메시지가 출력되며 다시 해독을 진행하지 않는다.

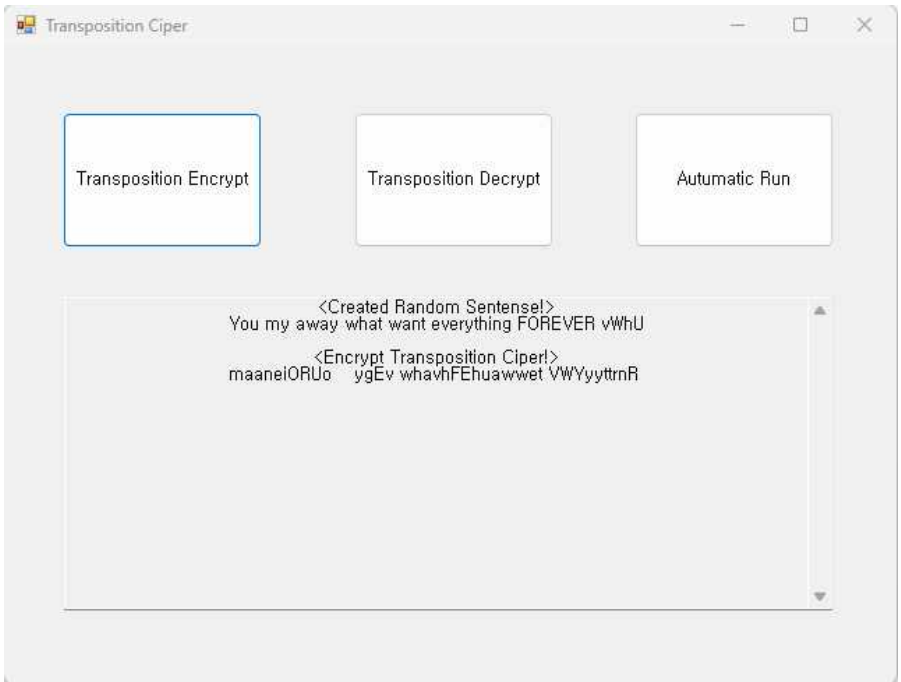


- Automatic Run 버튼을 클릭하여 시저 암호를 이용한 암호화와 해독을 100회 진행된 모습이다.
- 스레드를 이용해 함수를 실행했기 때문에, 병렬연산으로 인하여 빠르게 암호화와 해독이 진행되었으며, GUI 또한 암호화와 해독 속도에 맞춰 정보 출력이 진행되는 모습을 확인할 수 있다.
- 상단의 결과를 보면 100회 암호화와 해독을 진행하는데 평균 0.01867초라는 빠른 속도가 나왔다.

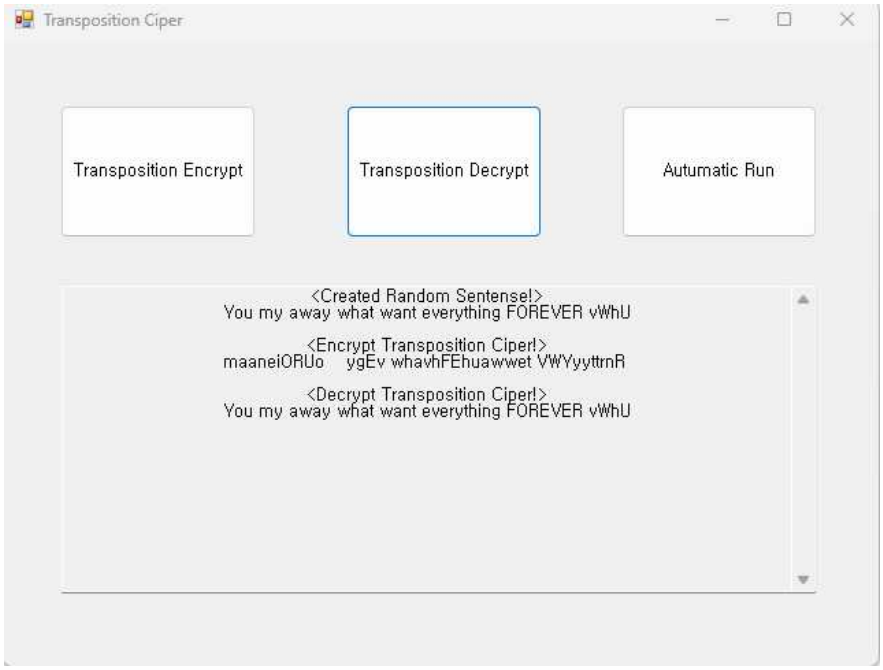
(2) 전치 암호



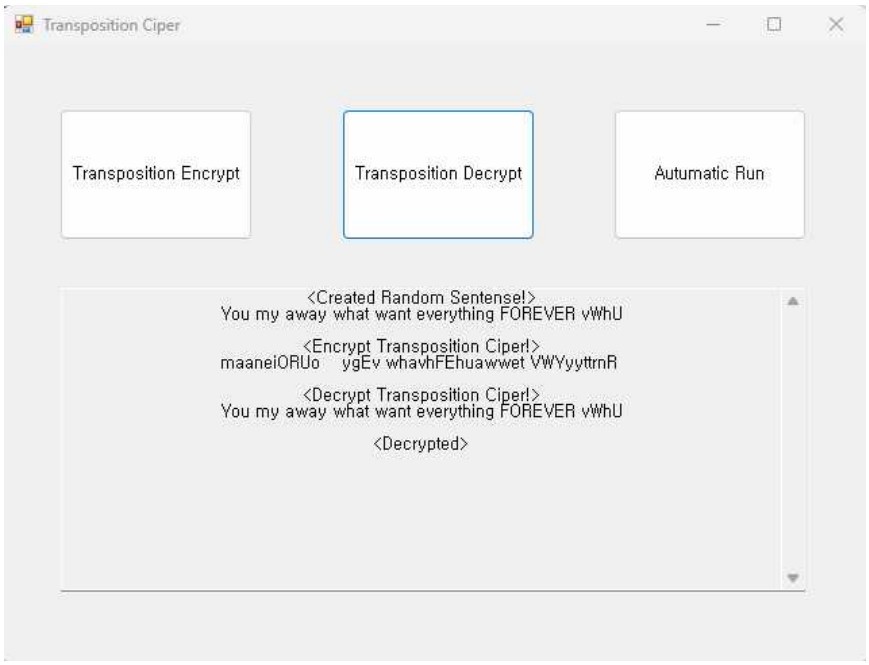
- 전치 암호를 이용해 암호화와 해독을 수행하는 프로그램의 GUI 모습이다.
- Transposition Encrypt 버튼을 클릭하면 문장 하나를 무작위로 생성하고, 생성된 문장의 암호화를 수행한다.
- Transposition Decrypt 버튼을 클릭하면 앞서 생성된 암호문의 해독을 수행한다.
- Automatic Run 버튼을 클릭하면 시저 암호의 암호화와 해독을 100회 반복 수행하고, 평균 시간을 출력한다.



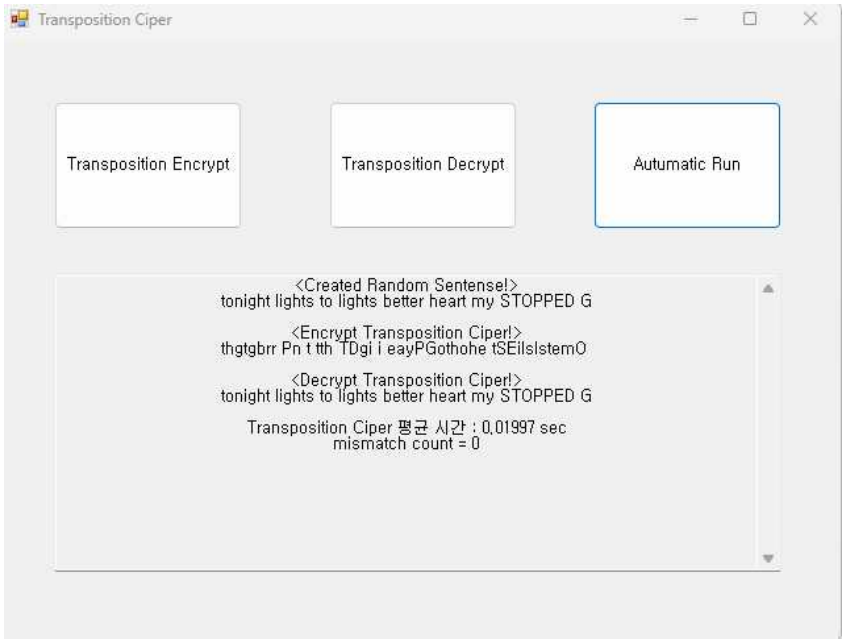
- Transposition Encrypt를 클릭한 모습이다.
- 처음에 You my away what want everything FOREVER vWhU라는 문장이 무작위로 생성되었으며, 이 문장을 전치 암호를 이용하여 암호화를 진행하였다.
- 생성된 암호문은 maaneiORUo ygEv whavhFEhuawwet VWYyyttrnR이다.
- 암호화를 진행하기 전에 행 단위로 읽으면서 작업 단위를 맞추기 위해 생성된 문장에 vWhU가 첨가된 모습을 볼 수 있다.
- 문장을 행 단위로 읽고 순서를 무작위로 바꾸어 열 단위로 표현했다.
- 문장 하나만 생성된 모습을 볼 수 있다.



- 앞서 생성된 암호문을 해독한 모습이다.
- 해독이 완료된 평문과 암호화를 하기 전 평문을 비교하면 완벽하게 해독된 모습을 볼 수 있다.
- 해독되는 원리는 암호화 과정을 역순으로 진행된 것으로, 암호문을 다시 열 단위로 다시 끊은 다음에, 자리바꿈 맵을 참조하여 문장들을 원래 순서대로 정렬하고 이를 행 단위로 읽어 복구한 것이다.



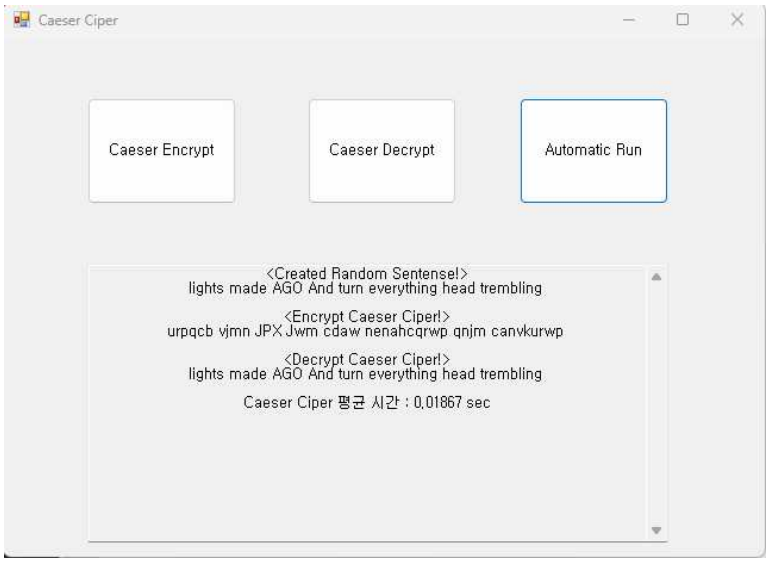
- 해독이 완료된 상태에서 다시 Transposition Decrypt 버튼을 클릭하면 해독이 완료되었음을 알리는 메시지가 출력되며 다시 해독을 진행하지 않는다.



- Automatic Run 버튼을 클릭하여 전치 암호를 이용한 암호화와 해독을 100회 진행된 모습이다.
- 스레드를 이용해 함수를 실행했기 때문에, 병렬연산으로 인하여 빠르게 암호화와 해독이 진행되었으며, GUI 또한 암호화와 해독 속도에 맞춰 정보 출력이 진행되는 모습을 확인할 수 있다.
- 상단의 결과를 보면 100회 암호화와 해독을 진행하는데 평균 0.01997초라는 빠른 속도가 나왔다.
- 해독한 평문과 암호화하기 전 평문이 다른 경우는 발생하지 않았다.

<결과 분석>

(1) 시저 암호



- 시저 암호를 이용한 암호화와 해독의 평균 소요 시간은 0.01867초가 나왔다.
- 소요 시간이 낮게 나온 이유는 영문자만을 대상으로 shift를 진행하였고 소문자는 소문자끼리, 대문자는 대문자끼리 shift를 진행하였기 때문에, 각 자리의 shift 횟수는 최대 25번이었다. 또한 각 문자가 shift 되는 횟수가 다르지 않고, 같이 shift 되기에 전체 경우의 수가 25개 밖에 존재하지 않아 소요 시간이 낮은 것으로 생각한다.

카이사르 암호 암호화/복호화

카이사르 암호란:
치환암호의 일종으로 로마의 황제 카이사르가
이 암호를 사용했다.
자세한 사항은 아래의 주소에서 확인 부탁드립니다.
https://highschoolfree.com/cry 사이트로 연결

ex) I love cookie

lights made AGO and turn everything head trembling

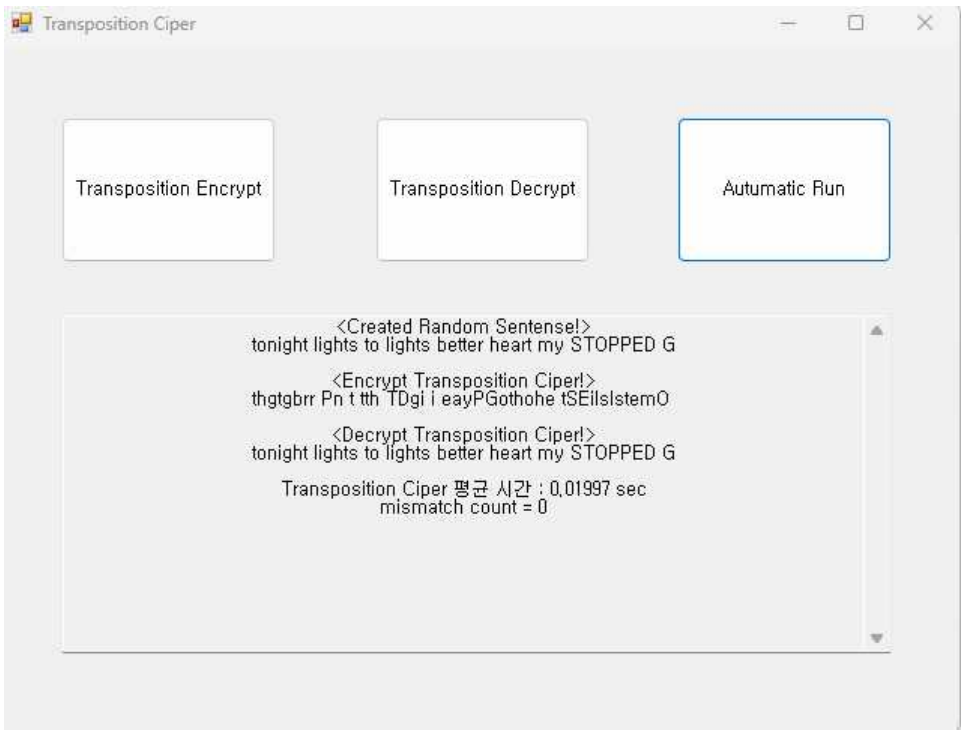
START

Answer is...

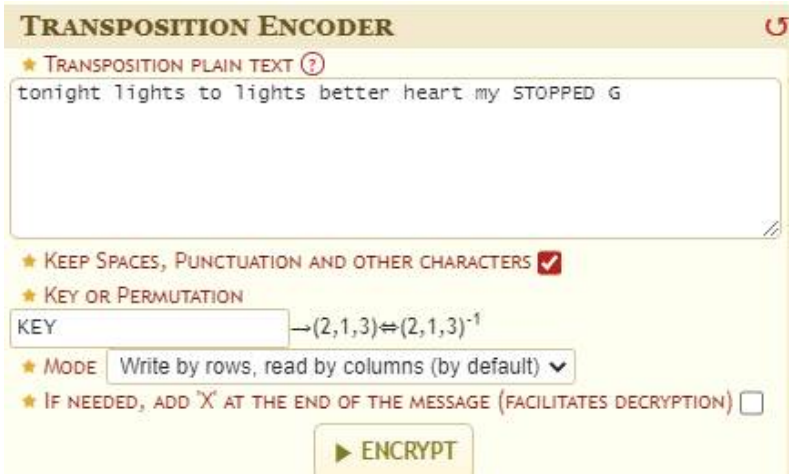
1. lights made AGO and turn everything head trembling
2. mjhiut nbef BHP boe uvso fwfszuijoh ifbe usfncmjoh
3. nkijvu ocfg CIQ cpf vwtp gxgtavjkpi jgcf vtgodnkpi
4. oljkwv pdgh DJR dqg wxuq hyhubwklqj khdg wuhpeolqj
5. pmklxw qehi EKS erh xyvr izivcxlmrk lieh xviqfpmrk
6. qnlmyx rfij FLT fsi yzws jajwdymnsl mjfi ywjrgqnsl
7. romnzy sgjk GMU gtj zaxt kbkxeznotm nkgj zxkshrotm
8. spnoaz thkl HNV huk abyu lclyfaopun olhk ayltispun
9. tqopba uilm IOW ivl bczv mdmzgbpqvo pmil bzmujtqvo
10. urpqcb vjmn JPX jwm cdaw nenahcqrwp qnjm canvkurwp
11. vsqrdc wkno KQY kxn debx ofobidrsxq rokn dbowlvsxq
12. wtrsed xlop LRZ lyo efcy pgpcjestyr splo ecpxmwtyr
13. xustfe ympq MSA mzp fgdz qhqdkftuzs tqmp fdqynxuzs
14. yvtugf znqr NTB naq ghea rirelguvat urnq gerzoyvat
15. zwuvhg aors OUC obr hifb sjsfmhvwbu vsor hfsapzwbu
16. axvwih bpst PVD pcs ijgc tktgniwxcv wtps igtbqaxcv
17. bywxji cqtu QWE qdt jkhd uluhajxydw xuqt jhucrbydw
18. czxykj druv RXF reu klie vmvipkyzex yvru kivdsczex

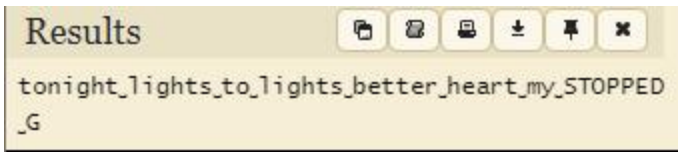
- 시저 암호의 암호화와 해독을 진행하는 다른 툴을 이용한 모습이다.
- 문장을 넣었을 때 하나의 문장에 대해서 0부터 26번의 shift가 진행된 문장, 즉 모든 경우의 수를 모두 출력하는 것을 볼 수 있다.
- 암호화와 해독을 진행하는 시간은 내가 구현한 프로그램이나 위의 툴이나 거의 비슷했다.
- 그러나 이번 실습의 목적인 말이 되는 단어와 문장의 후보군을 찾는 것에서 위의 툴은 매우 부적합하다. 내가 구현한 프로그램은 단어 셋이 따로 존재하기 때문에, 단어 셋과 문장에 포함된 단어를 비교해 후보군을 도출할 수 있지만, 위의 툴은 그런 비교 과정 없이 shift 된 모든 문장을 출력하기 때문이다.
- 그러나 반대로 단어 셋을 구현하는 경우 많은 단어를 저장해야 해서 프로그램이 무거워지고, 비교 과정까지 포함되기 때문에 속도 측면에서 손해를 볼 수 있다. 따라서 무조건 내 프로그램이 좋다, 위의 툴이 안 좋다고 말하기에는 부적절하고, 목적에 따라 취사해서 사용하는 것이 좋을 것이다.
- (https://jo-gunhee.github.io/website1/dcode/dcodewebsite.html)

(2) 전치 암호

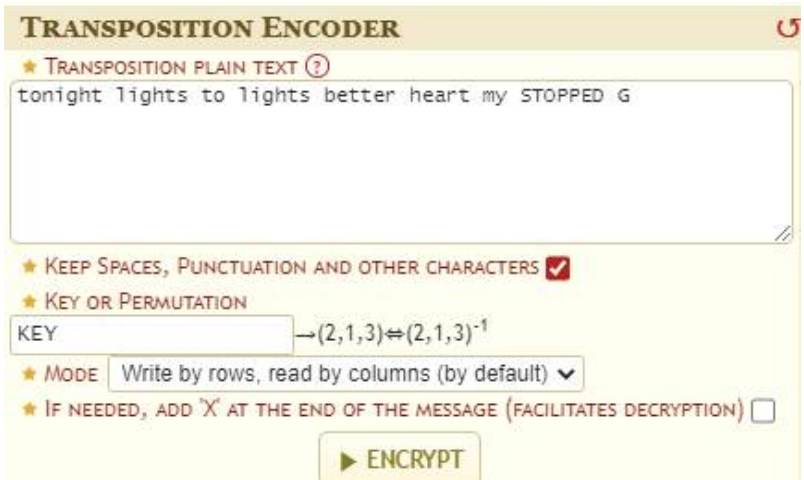


- 전치 암호를 이용한 암호화와 해독의 평균 소요 시간은 0.01997초가 나왔다.
- 소요 시간이 낮게 나온 이유는 평문을 행 단위로 읽는 과정에서 행을 5개로 나눠 읽었기 때문에 자리바꿈 맵을 구성하는데 원소가 5개 밖에 존재하지 않아 무작위로 섞고 열 단위로 읽는 과정이 어렵지 않았다.
- 반대로 만약 행을 여러 개로 나눠서 읽은 경우, 자리바꿈 맵에 차지하는 원소의 개수가 늘기 때문에 무작위로 섞는데 시간 복잡도가 증가할 것이고, 열 단위로 읽는 과정에서 참조를 많이 하게 되어 많은 시간이 소요될 것 같다.





- 전치 암호의 암호화와 해독을 진행하는 다른 툴을 이용한 모습이다.
- 옵션을 살펴보면 띄어쓰기, 구두점, 문자들을 그대로 유지하는 옵션과 key와 순열 조합을 선택하는 옵션, 모드 선택, 끝에 X를 추가하는 옵션이 존재한다.
- key를 선택하는 경우 문자열을 전치해서 처리하게 된다.
- 순열 조합을 선택하는 경우, 문자열의 순서를 무작위로 선택해서 암호화하게 된다.



- 암호문을 다시 평문으로 해독한 모습이다.
- 내가 구현한 프로그램이나 이 툴이나 소요 시간은 비슷하다.
- 암호화하기 전 평문과 암호문을 해독하여 얻은 평문을 비교하면 완벽하게 해독한 모습을 볼 수 있다.

- key 방법을 사용하는 경우 암호화가 비교적 간단하므로 금방 해독될 수 있지만, permutation의 경우에는 수많은 조합을 뚫고 하나의 조합을 찾아내야 해서 많은 시간이 소요될 것으로 생각한다.
- (https://www.dcode.fr/transposition-cipher)

3) 느낀 점

- 2주 동안의 brute force attack을 끝내고 이번에는 시저 암호와 전치 암호를 이용해 암호화를 진행해보고 해독해보는 과정을 실습해보았다. 시저 암호와 전치 암호는 암호화 기법 중 비교적 간단한 축에 속하기 때문에 부담 없이 프로그램을 구현할 수 있었다. 실습을 진행하면서 도중에 힘들었던 점이라면 전치 암호 프로그램을 구현할 때 벡터를 다루는 과정에서 out of range exception이 발생해 무슨 문제인지 알아보고 해결하는 것이었다. 간단한 문제였지만 너무 어렵게 생각해서 접근을 어렵게 했었던 것 같다. 벡터를 미리 초기화하고 벡터의 원소에 접근해야 하는데 reserve 함수로 공간만 예약한 상태에서 원소에 접근하니 오류가 계속 발생한 것이다. 다음부터는 벡터의 원소에 접근하려고 할 때 초기화가 되어있는지 확인하고 해야겠다. 그리고 시저 암호와 전치 암호 모두 예전에 어디선가 본 기억이 있었지만, 개념을 확실하게 알지 못했었는데 이번 실습에서 프로그램을 직접 구현해보면서 시저 암호와 전치 암호에 대해 확실하게 기억할 수 있었다. 그리고 전치 암호를 구현하면서 오랜만에 다양한 자료구조를 사용하게 되었는데, 자리바꿈 맵을 구현하기 위해 처음에 map 라이브러리를 사용해서 구현했다가 map에서 random_shuffle을 사용할 수 없다고 오류가 뜨는 것이었다. 평소에 map을 잘 사용하지 않으니 이런 점을 잘 알지 못했고, 이번 실습을 진행할 때 알게 된 것이었다. 그래서 pair<int, string>를 이용해서 벡터로 구성해 맵을 구성하고, 벡터를 random_shuffle 했다. 아무래도 c++을 사용하면서 평소에 자주 사용하던 자료구조만 사용하다 보니 다른 자료구조에 대한 특징을 많이 잊어버렸었다. c++을 c처럼 사용하는 게 아닌지 생각해보게 되었고, 다음부터는 c++의 다양한 기능을 사용할 수 있도록 노력해야겠다.