1. Template를 사용하여 오름차순으로 정렬하는 함수를 만들어주세요. 문자열의 경우에는 사전 순입니다.(algorithm 라이브러리의 sort 함수 사용금지)

1) sort함수와 print함수는 template를 활용하세요. 2) print함수는 iterator를 활용하여 출력하세요. 3) 문자열 정렬의 경우에는 사전 순으로 출력됩니다.

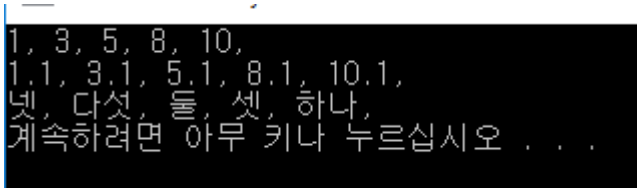
```
int main() {
		vector<int> int_list(5);
		int_list = { 10, 5, 8, 1, 3 };
		vector<double> double_list(5);
		double_list = { 10.1, 5.1, 8.1, 1.1, 3.1 };
		vector<string> string_list(5);
		string_list = { "하나", "둘", "셋", "넷", "다섯" };

//sort, print함수는 매개변수 오버로딩이 아닌 template를 활용하여 만드세요.
//print함수는 iterator를 활용하세요
		sort(int_list);
		sort(double_list);
		sort(string_list);

		print(int_list);
		print(double_list);
		print(string_list);

}
```

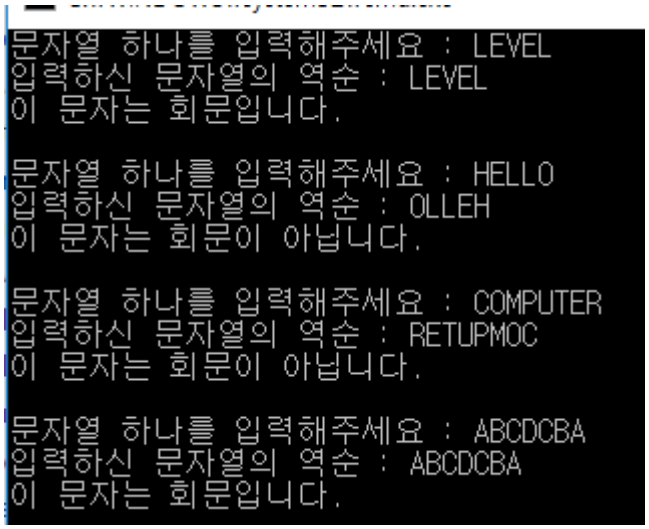
1 – 출력화면



2. 람다 함수를 활용하여 회문을 판별하는 프로그램을 작성하세요.

1) 단어를 뒤집어도 똑같은 단어를 회문이라고 정의합니다.(ex. level) 2) 람다 함수를 활용하여 회문을 판별하는 프로그램을 작성하세요.

2 – 출력 예시



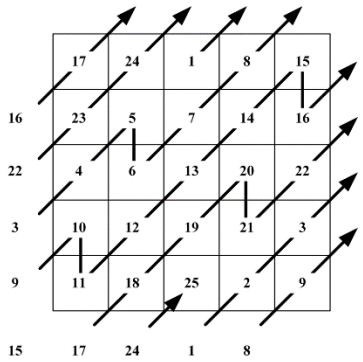
3. 홀수 숫자 n을 하나 입력받고, n*n 크기의 마방진을 출력하는 프로그래밍을 작성하세요.

1) 마방진이란, n*n 행렬에서 가로, 세로, 대각선 방향의 숫자를 더하면 모두 같은 값이 나오는 배열입니다.

- 예시

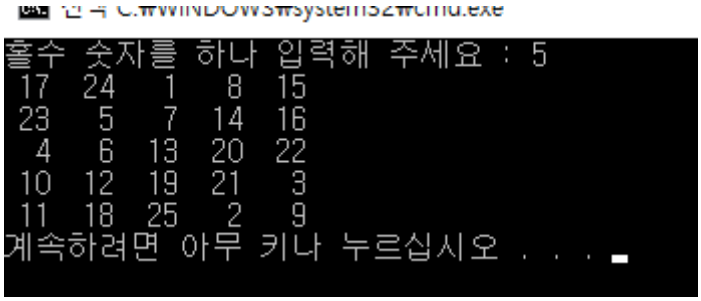
4	9	2
3	5	7
8	1	6

2) 마방진을 만드는 원리는 1에서부터(보통 1은 첫 번째 줄 가운데에 두고 시작합니다.) 오른쪽 위 대각선 방향으로 숫자를 하나씩 늘려가는 방식을 사용합니다.



3 - 출력예시

```
홀수 숫자를 하나 입력해 주세요 : 3
  8   1   6
  3   5   7
  4   9   2
계속하려면 아무 키나 누르십시오 . . .
```



4. 다양한 Type을 사용하는 Queue Class를 구현하세요

변수(private):

T* p_list; // 정수형 변수들을 가지는 배열

int size; //현재 저장된 변수들의 개수

int MAX_SIZE; // 최대로 저장할 수 있는 p_list의 크기

함수(public);

Queue(int _MAX_SIZE = 1000) //생성자: p_list의 크기를 MAX_SIZE만큼 동적 할당.

~Queue() // 소멸자: p_list의 동적 할당을 해제

int find_index(T _item) // p_list에서 _item과 동일한 값이 있는지 검색 후 발견시 index를 반환한다 만약 발견하지 못하면 -1을 반환한다

void Enqueue(T _item) // 입력item을 p_list의 끝에 저장한다. 만약 _item과 동일한 값이 p_list에 존재할 경우 p_list에 _입력 item을 추가하지 않는다. (힌트: find_index 함수를 사용해서 중복된 값이 p_list에 있는지 조사후 없는 경우에 입력 item을 p_list에 추가). size가 MAX_SIZE보다 크면 item을 추가하지 않는다.("Error: out of memory"출력)

T Dequeue() // p_list에 있는 첫번째 item을 제거한다음 그 아이템을 return한다 (힌트:size 값을 줄이면 p_list의 아이템을 제거한 것과 동일한 효과) size가 0일 때는 item을 제거하지 않는다. ("Error: No item exists in the list"출력)

void print() const // Queue 객체의 item들을 출력한다

int get_size() //Queue 객체의 크기를 출력한다

T get_item(int _index) // p_list의 해당 index에 있는 item 값을 리턴한다.

시작 코드

```
#do not modify below

int main()

{

	Queue<int> int_queue;

	Queue<float> float_queue;

	Queue<char> char_queue;


	int_queue.Enqueue(1);

	int_queue.Enqueue(2);

	int_queue.Enqueue(2);

	int_queue.Enqueue(5);


	float_queue.Enqueue(4.3);

	float_queue.Enqueue(2.5);

	float_queue.Enqueue(3.7);

	float_queue.Enqueue(3.7);


	char_queue.Enqueue('b');

	char_queue.Enqueue('b');

	char_queue.Enqueue('c');

	char_queue.Enqueue('a');


	cout << "<Before Dequeue>" << endl;

	int_queue.print();

	float_queue.print();

	char_queue.print();
```

```
        int_queue.Dequeue();
        float_queue.Dequeue();
        float_queue.Dequeue();
        char_queue.Dequeue();
        char_queue.Dequeue();
        char_queue.Dequeue();
        char_queue.Dequeue();

        cout << "<After Dequeue>" << endl;
        int_queue.print();
        float_queue.print();
        char_queue.print();

        return 0;
}
```

```
<Before Dequeue>
Items in the list : 1, 2, 5,
Items in the list : 4.3, 2.5, 3.7,
Items in the list : b, c, a,
Error: No item exist in the list
<After Dequeue>
Items in the list : 2, 5,
Items in the list : 3.7,
Items in the list :
```