Practice 11

- 실습문제 1: 동적 메모리 할당 기초

배열의 길이를 사용자로부터 입력 받아라. 그런 다음 동적 메모리 할당을 이용하여 입력 받은 길이 크기의 Integer 배열을 만들고 사용자 입력을 통해 배열의 내용을 채워라.

```
int main(){
	int Dsize;
	cout << "Enter a size of the array : ";
	cin >> Dsize;
	// (코드 작성) Dsize 크기의 동적 배열 할당. Integer 형
	// (코드 작성) 사용자 입력으로 배열 채우기(포인터 간접 연산자 활용)
	// (코드 작성) 배열 출력하기(포인터에 배열 첨자 연산자 활용)
	// (코드 작성) 동적 배열 메모리 해제
	return 0;
}
```

Practice 11. Advanced Pointer

- ## 실습문제 2: Pointer Misuse

다음 두 코드는 메모리를 잘못 사용하는 대표적인 예이다. 첫 번째 코드는 dangling pointer 의 예이고 두 번째 코드는 memory leak 의 예이다. 코드를 돌려보고 어떤 결과가 나오는지 확인하라. 그리고 왜 잘못된 코드인지 comment 형식으로 파일 안에 쓰라. 그리고 잘못된 코드를 고쳐라.

```
//렌덤 넘버가 짝수이면 e를 출력하고 홀수이면
//o를 출력하는 프로그램. Answer라는 변수의 scope에 주목
int main()
{
	srand(time(NULL));
	char* ptr;
	int randomNumber = (rand() % 100) + 1;
	if((randomNumber%2) == 0) {//If the number is even
		char answer = 'e';
		ptr = &answer;
	} else {//if the number is odd
		char answer = 'o';
		ptr = &answer;
	}
	cout << *ptr;
}
```

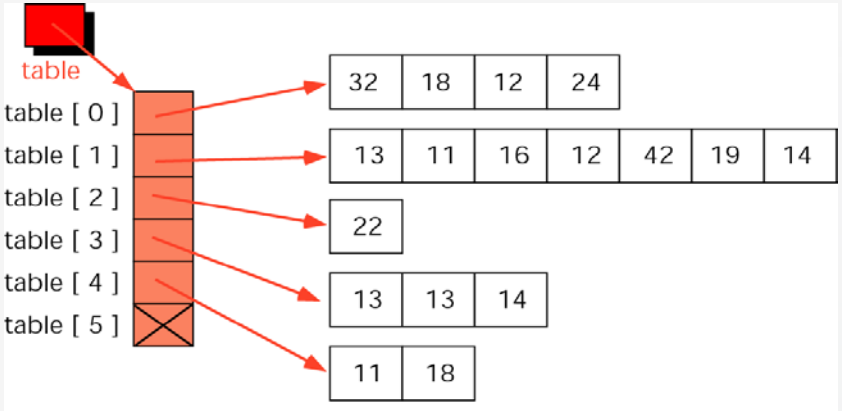
```
//사용자 입력을 받아 입력한 수 만큼 자기 이름을
//프린트 해주는 프로그램. 프로그램은 돌아가지만
//memory leak 이 발생
void printYourName() {
	char* myName = new char[20];
	strcpy(myName,"Your Name");
	cout << myName << endl;
}

int main() {
	int input;
	cout << "how many times would you like to print
your name?";
	cin >> input;
	for(int a = 0; a < input; a++) {
		printYourName();
	}
	return 0;
}
```

Practice 11. Advanced Pointer

- ## 실습문제 3: Ragged Array

다음 그림과 같이 불규칙한 크기의 배열을 저장할 수 있는 배열을 만들고 아래 그림과 같이 내용을 초기화 하라. 그런 다음 아래 출력 예와 같이 출력하라. (수업자료 참조)




```
32 18 12 24
13 11 16 12 42 19 14
22
13 13 14
11 18
```

Practice 11. Advanced Pointer

- ## 실습문제 4: Linked List 1

Integer를 데이터로 가지는 Single linked list를 수업 시 배포한 소스코드를 참조하여 구현하고 다음과 같은 함수를 추가 구현하라.

int removeDuplicate(NodePtr head);

위 함수는 리스트 안에 중복되어 있는 값(노드)을 없애주는 함수이다. 예를 들어 10이라는 값을 가지는 노드가 3개 있으면 첫 번째 노드만 남기고 뒤 두 개를 지워준다. 몇 개의 노드가 지워졌는지 int형으로 리턴한다. head가 가리키는 첫 번째 노드는 절대 지워지지 않게 알고리즘을 구현하라.

```
typedef int ItemType;
struct NodeType;
typedef NodeType* NodePtr;
struct NodeType {
	ItemType component;
	NodePtr link;
};
int removeDuplicate(NodePtr head);
int main()
{
	NodePtr head;
	//[사용자에게 몇 개의 노드를 가지는 리스트를 만들지 입력 받음]
	//[그 숫자만큼 1~20 렌덤넘버로 head가 가리키는 리스트를 채움]
	//[리스트 출력]
	cout << removeDuplicate(head) << " nodes removed\n";
	//[리스트 출력]
	return 0;
}
int removeDuplicate(NodePtr head) {
	//[구현]
}
```

```
Size of nodes? 5
A list with 5 nodes generated:
List: 3 10 15 2 3
1 nodes removed
List: 3 10 15 2
```

Practice 11. Advanced Pointer

- ## 실습문제 5: Linked List 2

실습문제 4번에 다음 함수를 추가하여 구현하라.

bool insertList(NodePtr head, NodePtr headInsert, int afterThis);

head라는 리스트 안에서 afterThis라는 값을 가지고 있는 노드 바로 뒤에 headList 라는 또 다른 리스트를 삽입하는 함수이다. afterThis라는 데이터를 찾지 못하면 false를 리턴하고 삽입이 성공하면 true를 리턴한다.

```
typedef int ItemType;
struct NodeType;
typedef NodeType* NodePtr;
struct NodeType {
	ItemType component;
	NodePtr link;
};
bool insertList(NodePtr head, NodePtr headInsert, int afterThis);

int main()
{
	NodePtr head;
	…	//실습문제 4번과 동일
	…
	cout << removeDuplicate(head) << " nodes removed\n";
	//[리스트 출력]

	NodePtr headInsert;
	//[1~20사이의 렌덤한 값으로 채워진 사이즈가 10인 리스트 생성]
	if(insertList(head,headInsert,10)) {
		cout << "Insertion successful\n";
	} else {
		cout << "Insertion failed\n";
	}
	//[최종 head 리스트의 값을 출력]
	return 0;
}
bool insertList(NodePtr head, NodePtr headInsert, int afterThis) {
	//[구현]
}
```

```
Size of nodes? 5
A list with 5 nodes generated:
List: 3 10 15 2 3
1 nodes removed
List: 3 10 15 2
Insertion successful
List: 3 10 8 4 5 10 12 17 19 2 1 10 15 2
```

- 실습문제 6: Doubly-Linked List

더블 링크드 리스트를 구현하고 다음 함수를 구현하라.

bool addBefore(NodeType* head, int beforeThis, int addThis)

head라는 리스트 안에서 beforeThis라는 값을 가지고 있는 노드 바로 앞에 addThis 라는 노드를 삽입하는 함수이다. beforeThis라는 데이터를 찾지 못하면 false를 리턴하고 삽입이 성공하면 true를 리턴한다.

노드 Structure는 다음과 같음

```
struct NodeType{
    int payload;
    NodeType *prev;
    NodeType *next;
};
```

더블 링크드 리스트 예