

# Object Oriented Programming assignment3

설계 분반: 2 (신동화 교수님)

실습 분반: 1 (신동화 교수님)

학번: 2023202032

이름: 남호준

1.

## 문제 설명

1번 문제는 원형 연결 리스트를 구성하고, 값 전송, 리스트 출력, 종료를 수행하는 프로그램이다. 값 전송 시엔 10% 확률로 연결이 끊어진다. 끊어질 시에는 오류 메시지와 끊어진 부분은 출력하고 프로그램을 종료한다.

## 알고리즘

### Algorithm 1 Node Class

```
1: Properties:
2:   int data
3:   int index
4:   Node* next
5: Methods:
6: Constructor(data, index):
7:   Initialize data, index, next
8: Destructor():
9:   Reset data, index, next
10: SetNext(Node* next):
11:   Set next pointer
12: SetData(int data):
13:   Set data
14: GetData() -> int:
15:   Return data
16: GetIndex() -> int:
17:   Return index
18: GetNext() -> Node*:
19:   Return next pointer
```

### Algorithm 2 List Class

```
1: Properties:
2:   Node* head
3:   Node* tail
4:   Node* tmp
5:   Node* pre
6: Methods:
7: Constructor():
8:   Initialize head, tail, tmp, pre to null
9: Destructor():
10:  if tail is null then
11:    Set tmp to head
12:  else
13:    Set tmp to tail
14:  end if
15:  while tmp is not null do
16:    Set pre to tmp
17:    Set tmp to tmp.getnext()
18:    Delete pre
19:  end while
20: Initialize(int[] val):
21:  for i = 0 to 9 do
22:    Create new node with val[i], i
23:    if head is null then
24:      Set head to new node
25:    else
26:      Set tmp to head
27:      for j = 0 to i-1 do
28:        Set tmp to tmp.getnext()
29:      end for
30:      Set tmp.next to new node
31:      Set new node.next to null
32:    end if
33:  end for
34:  Set tmp to head
35:  while tmp.getnext() is not null do
36:    Set tmp to tmp.getnext()
37:  end while
38:  Set tmp.next to head
39: Transfer(int id1, int id2) -> bool:
40:  Set val to 0
41:  Set tmp to head
42:  while tmp.getindex() is not id1 do
43:    Set tmp to tmp.getnext()
44:  end while
45:  Set val to tmp.getdata()
46:  while tmp.getindex() is not id2 do
47:    if probabilistic_disconnected_func() then
48:      Set tail to tmp.getnext()
49:      Print "Detected a disconnection between tmp.getindex() and
      tmp.getnext().getindex()"
50:      Set tmp.next to null
51:      return true
52:    end if
53:    Set tmp to tmp.getnext()
54:  end while
55:  Set tmp.data to val
56:  return false
57: Print():
58:  Set tmp to head
```

---

**Algorithm 3** Main Function

---

```
1: Create list object a
2: Initialize command array with "initialize", "transfer", "print", "exit"
3: Initialize val array with 10 elements
4: Initialize id1, id2, sub_input, input
5: while true do
6:   Print "Command: "
7:   Read input
8:   if input is "initialize" then
9:     for i = 0 to 9 do
10:      Read val[i]
11:    end for
12:    Call a.initialize(val)
13:  else if input is "transfer" then
14:    Read id1, sub_input, id2
15:    if sub_input is not "to" then
16:      end if
17:    if a.transfer(id1, id2) then
18:      return 0
19:    end if
20:  else if input is "print" then
21:    Call a.print()
22:  else if input is "exit" then
23:    end if
24: end while
25: return 0
```

---

확률 절단 함수는 0일 때 끊어지도록 true를 반환한다.

transfer 명령 시 id1을 찾아 리스트 이동, 데이터 저장, id2를 찾아 리스트 이동, 데이터 대입으로 행동한다. 이때, 데이터를 저장 후 이동할 때 확률 절단 함수가 이동 시마다 실행된다.

## 결과 화면

```
Command : initialize 1 2 3 4 5 6 7 8 9 10
Command : print
Output   : 1 2 3 4 5 6 7 8 9 10
Command : transfer 1 to 9
Output   : Detected a disconnection between 4 and 5
```

초기화 한 값을 프린트 했을 때, 출력된다.

transfer 커맨드를 사용하여 인덱스의 1에 있는 2의 값을 인덱스 9에 있는 10에 전송하는 도중, 4와 5 인덱스 사이에서 끊어져 프로그램이 종료됐다.

## 고찰

1번 문제에서 헛갈리는 부분이 많아 해맨 적이 많다.

원형 연결 리스트를 처음 만들면서 마지막 노드를 잇고, 연결 리스트를 순회하는데 조건을 잘못 설정하여 무한 루프에 빠지거나, 값이 들어가지 않는 등 여러 문제가 있었다. 또한, 확률적으로 disconnect하는 함수가 실행되는 부분을 잘못 설정하여 데이터를 가져오기 전, 즉 전송 이전에 끊어져 잘못되는 부분도 있었다. 문제에 조건을 못 읽은 부분도 있었지만, 조건이 좀 더 자세했으면 원활하게 진행되었을 것이라 생각한다.

2.

## 문제 설명

2번 문제는 행렬 클래스를 만들어 데이터 세팅, 출력, 연산자 오버로딩을 통한 행렬 연산을 지원하는 프로그램을 만드는 문제이다.

이때, 행렬과 상수값을 연산할 수 있는 연산자 오버로딩도 포함되어 있다.

행렬과 행렬은  $+, -, *$ 을 지원, 행렬과 상수는  $+, -, *, /$  연산을 지원하게 만들어야 한다.

## 알고리즘

---

### Algorithm 1 Matrix Class

---

```
1: Properties:
2:   double data[4][4]
3: Methods:
4: Constructor():
5:   Initialize all elements of data to 0
6: Destructor():
7:   Reset all elements of data to 0
8: Operator+(Matrix mtx) -> Matrix:
9:   Create a new Matrix
10:  for each element in data do
11:    Add corresponding elements of this and mtx
12:  end for
13:  Return the new Matrix
14: Operator-(Matrix mtx) -> Matrix:
15:   Create a new Matrix
16:  for each element in data do
17:    Subtract corresponding elements of mtx from this
18:  end for
19:  Return the new Matrix
20: Operator*(Matrix mtx) -> Matrix:
21:   Create a new Matrix
22:  for each element in data do
23:    Multiply corresponding elements and sum for matrix multiplication
24:  end for
25:  Return the new Matrix
26: Operator+(int Scalar) -> Matrix:
27:   Create a new Matrix
28:  for each element in data do
29:    Add Scalar to element
30:  end for
31:  Return the new Matrix
32: Operator-(int Scalar) -> Matrix:
33:   Create a new Matrix
34:  for each element in data do
35:    Subtract Scalar from element
36:  end for
37:  Return the new Matrix
38: Operator*(int Scalar) -> Matrix:
39:   Create a new Matrix
40:  for each element in data do
41:    Multiply element by Scalar
42:  end for
43:  Return the new Matrix
44: Operator/(int Scalar) -> Matrix:
45:   Create a new Matrix
46:  for each element in data do
47:    Divide element by Scalar
48:  end for
49:  Return the new Matrix
50: setValue(double arr[4][4]):
51:  for each element in arr do
52:    Copy value to data
53:  end for
54: display():
55:  for each element in data do
56:    Print element
57:  end for
```

---

---

**Algorithm 2** Main Function

---

```
1: Create Matrix object mat1
2: Create Matrix object mat2
3: Initialize arr1 with values:
4:   { {0,0,0,0}, {1,1,1,1}, {2,2,2,2}, {3,3,3,3} }
5: Initialize arr2 with values:
6:   { {0,1,2,3}, {0,1,2,3}, {0,1,2,3}, {0,1,2,3} }
7: Set values of mat1 using arr1
8: Set values of mat2 using arr2
9: Create Matrix object mat3 as the sum of mat1 and mat2
10: Print "Matrix 1:"
11: Call mat1.display()
12: Print "Matrix 2:"
13: Call mat2.display()
14: Print "Matrix 3:"
15: Call mat3.display()
16: return 0
```

---

**결과 화면**

```
Matrix 1:
0 0 0 0
1 1 1 1
2 2 2 2
3 3 3 3

Matrix 2:
0 1 2 3
0 1 2 3
0 1 2 3
0 1 2 3

Matrix 3:
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```

예시 화면과 같은 main code를 작성하였고, 같게 나왔다.

**고찰**

연산자 오버로딩을 처음 하면서 어떻게 해야 할지 감이 안 왔지만, 하나를 구현하니 나머진 연산 부분 빼곤 다 같아서 금방 해결할 수 있었다.

3.

### 문제 설명

3번 문제는 멤버십 파일을 이용하고, 리스트를 관리하여 로그인 프로그램을 만드는 것이다. 파일을 이용해서 프로그램이 종료되어도 리스트를 저장하고 불러와 정보를 유지해야 한다.

프로그램은 로그인, 등록, 탈퇴, 종료 명령을 수행한다.

암호는 숫자, 문자, 특수문자를 포함해야 한다. 로그인할 시 탈퇴할 수 있다. 비밀번호는 암호화하여 저장하여야 한다. 암호화는 시저 암호화를 이용해 아이디 길이만큼 비밀번호의 영문자를 오른쪽으로 밀어 암호화한다.

### 알고리즘

Algorithm 1 Membership Class

```
1: Properties:
2:   Member* list
3:   int cnt
4: Methods:
5: Constructor():
6:   Initialize cnt to 0
7:   If member_list.txt exists:
8:     Read all members from file into list
9:     Increment cnt for each member
10: Destructor():
11:   Write all members from list to member_list.txt
12: login_correct(Member a) -> bool:
13: for each member in list do
14:   if id matches then
15:     Decode the member's password
16:     if password matches then
17:       return true
18:   end if
19: end if
20: end for
21: return false
22: register_member(Member a) -> bool:
23: if id already exists in list or password is invalid then
24:   return false
25: end if
26: Encode the password and add the member to list
27: Increment cnt
28: return true
29: ps_correct(Member a) -> bool:
30: Check if password meets criteria:
31:   Length between 10 and 20
32:   Contains letters, numbers, and special characters
33: return true if valid, false otherwise
34: withdraw(Member a):
35: for each member in list do
36:   if id matches then
37:     Remove the member
38:   end if
39: end for
40: Encoding(Member a) -> Member:
41: Encode the member's password using Caesar cipher
42: decoding(Member a) -> Member:
43: Decode the member's password using Caesar cipher
```

Algorithm 2 Main Function


```
1: Initialize membership object a
2: Initialize input_user, login_user, login_flag
3: while true do
4:   Display menu
5:   if login option selected then
6:     Get id and password from user
7:     if login_correct then
8:       Set login_flag to true
9:     else
10:      Display login failed message
11:    end if
12:  else if register option selected then
13:    Get id and password from user
14:    if register_member then
15:      Display success message
16:    else
17:      Display invalid message
18:    end if
19:  else if withdraw option selected then
20:    if login_flag is false then
21:      Display invalid message
22:    else
23:      Call withdraw
24:      Reset login_user and login_flag
25:    end if
26:  else if exit option selected then
27:    break loop
28:  end if
29: end while
```



## 결과 화면

```
=====
Menu.
1. Login
2. Register
3. withdraw
4. exit
   : 2
-----
User id: hojun3338
password: mokwang0116
-----
Invalid.
=====
Menu.
1. Login
2. Register
3. withdraw
4. exit
   : 2
-----
User id: hojun3338
password: mokwang0116
-----
Invalid.
=====
```

```
=====
Menu.
1. Login
2. Register
3. withdraw
4. exit
   : 2
-----
User id: hojun3338
password: mokwang0116!
-----
Invalid.
=====
Menu.
1. Login
2. Register
3. withdraw
4. exit
   : 1
-----
User id: hojun3338
password: mokwang0116!
-----
Login successful.
=====
Logged in user : (hojun3338)
Menu.
1. Login
2. Register
3. withdraw
4. exit
   : 4
-----
C:\Users\hojun\OneDrive\바탕
이(가) 종료되었습니다(코드 :
```

 member\_list.txt

파일   편집   보기

hojun3338  
vxtfjwp0116!  
minjae  
gyjwck123!

파일 내에 있던 데이터에서 가져와 첫 번째 사진처럼 실행하였다.  
다시 프로그램을 실행하여 오른쪽 화면처럼 실행하였다.

### 고찰

데이터를 관리하는 부분에서 문제가 많이 생겼다.  
새로운 멤버를 여러 번 추가하는 부분을 나중에 깨달았는데, 리스트를 추가할 때마다 동적 메모리를 관리해야 하기 때문에 연결 리스트 또는 배열 복사를 이용해야 했는데 동적 배열 복사를 이용해서 메모리를 관리했다. 또한, 파일에 저장할 때, 아이디와 비밀번호를 각각 한 줄씩 적어 두줄이 하나의 멤버를 가리키도록 만들었다.

4.

### 문제 설명

4번 문제는 세계 각지의 시간을 출력하는 프로그램이다.  
그리니치 천문대를 기준으로 표준 시간을 정한다.  
세팅, 시간 더하기, 출력, 종료 명령을 수행한다.

### 알고리즘

먼저, 그리니치 천문대를 기준으로 워싱턴DC는 -5시간, 파리는 +7시간, 한국은 +8시간이다. 이때, 날짜에 관한 언급은 없으므로 24시간이 넘어가면 24로 나누어 나머지만 남겼다.

아래는 슈도코드이다.

Class Definitions

Class Time

Algorithm 1 Class Time

```
1: Attributes:
2:   int hour
3:   int minute
4:   int second
5: Constructor Time(hour, minute, second)
6:   Initialize hour, minute, second
7: Destructor Time()
8:   Reset hour, minute, second to 0
9: Method setTime(hour, minute, second)
10:   Set hour, minute, second
11: Method addTime(seconds)
12:   Add seconds to current time
13:   If seconds ≥ 60:
14:     Update minute and second
15:   If minutes ≥ 60:
16:     Update hour and minute
17:   If hour ≥ 24:
18:     Update hour
19: Method printTime()
20:   Print time in HH:MM:SS format
```

Class Korea

Algorithm 2 Class Korea inherits Time

```
1: Constructor Korea(hour, minute, second)
2:   Call parent constructor with hour, minute, second
3: Destructor Korea()
```

Class WashingtonDC

Class Paris

Class GreenwichObservatory

Algorithm 3 Class WashingtonDC inherits Time

```
1: Constructor WashingtonDC(hour, minute, second)
2:   Call parent constructor with hour, minute, second
3: Destructor WashingtonDC()
```

Algorithm 4 Class Paris inherits Time

```
1: Constructor Paris(hour, minute, second)
2:   Call parent constructor with hour, minute, second
3: Destructor Paris()
```

Algorithm 5 Class GreenwichObservatory inherits Time

```
1: Constructor GreenwichObservatory(hour, minute, second)
2:   Call parent constructor with hour, minute, second
3: Destructor GreenwichObservatory()
```

Algorithm 6 Main Function

```
1: Create Korea, WashingtonDC, Paris, GreenwichObservatory objects with
   initial time 00:00:00
2: while true do
3:   Print "Command : "
4:   Read input command
5:   if command is "setting" then
6:     Get current time in seconds
7:     Seed random number generator
8:     Generate random hour, minute, second
9:     Set Greenwich time with random values
10:    Calculate and set WashingtonDC time
11:    Calculate and set Paris time
12:    Calculate and set Korea time
13:   else if command is "add" then
14:     Read seconds to add
15:     Add seconds to all time objects
16:   else if command is "print" then
17:     Print times of Korea, WashingtonDC, Paris, and GreenwichObservatory
18:   else if command is "exit" then
19:     Break the loop
20:   end if
21: end while
22: Return 0
```



## 결과 화면

```
Command : setting
Command : print

Output :
Korea           = 08:33:57
WashingtonDC    = 19:33:57
Paris           = 01:33:57
GreenwichObservatory = 00:33:57

Command : add 3
Command : print

Output :
Korea           = 08:34:00
WashingtonDC    = 19:34:00
Paris           = 01:34:00
GreenwichObservatory = 00:34:00

Command : add 10000
Command : print

Output :
Korea           = 11:20:40
WashingtonDC    = 22:20:40
Paris           = 04:20:40
GreenwichObservatory = 03:20:40

Command : exit
```

시간을 한번 세팅하고 출력하였다.

그리니치 천문대를 기준으로 위에 적은 대로 시간을 설정하여 더하였다.

add 3 이후 출력으로 시간이 넘어감을 보였다.

이후 add 10000으로 시간이 넘어감을 보였다.

## 고찰

그리니치 표준시가 계절에 따라 바뀌기도 한다는 이야기를 듣고 문제가 있었다. 구현한 시차가 오류가 있어 수정하였다. 시간 세팅은 각각 %연산자와 시간이 넘어갔을 때, 넘어간 몫만큼 다음 시, 분으로 넘기고 %연산자로 남은 수를 계산하였다.

5.

문제 설명

5번 문제는 다항식을 만드는 클래스를 구현하는 프로그램을 만드는 것이다. 다항식은 0 이상의 x 차수를 가지고, 정수 계수를 가진다. 연산자 오버로딩을 통해 다항식을 더할 수 있다. 그리고 다항식의 변수 x에 값을 넣은 식의 값을 계산하는 함수, 다항식을 미분하는 함수를 구현해야 한다.

알고리즘

Class Definitions

Class Term

Algorithm 1 Class Term	
1:	<b>Attributes:</b>
2:	int m_Coefficient
3:	int m_Exponent
4:	Term* m_pNext
5:	<b>Constructor Term(coefficient, exponent)</b>
6:	Initialize m_Coefficient with coefficient
7:	Initialize m_Exponent with exponent
8:	Initialize m_pNext with NULL
9:	<b>Destructor Term()</b>
10:	<b>Methods:</b>
11:	getNext(): Return m_pNext
12:	getCoe(): Return m_Coefficient
13:	getExp(): Return m_Exponent
14:	setCoe(coefficient): Set m_Coefficient to coefficient
15:	setNext(pNext): Set m_pNext to pNext

Class Polynomial

---

**Algorithm 2** Class Polynomial

---

```
1: Attributes:
2:   Term* m_pHead
3: Constructor Polynomial()
4:   Initialize m_pHead with NULL
5: Destructor Polynomial()
6:   Set tmp to m_pHead
7:   While tmp is not NULL:
8:     Set cur to tmp
9:     Set tmp to tmp.getNext()
10:   Delete cur
11: Copy Constructor Polynomial(poly)
12:   Set p to poly.m_pHead
13:   While p is not NULL:
14:     Call addTerm with p.getCoe(), p.getExp()
15:     Set p to p.getNext()
16: Assignment Operator Polynomial=(poly)
17:   If this is equal to poly, return this
18:   Set tmp to m_pHead
19:   While tmp is not NULL:
20:     Set cur to tmp
21:     Set tmp to tmp.getNext()
22:   Delete cur
23:   Set p to poly.m_pHead
24:   While p is not NULL:
25:     Call addTerm with p.getCoe(), p.getExp()
26:     Set p to p.getNext()
27:   Return this
28: Method addTerm(coeff, exp)
29:   Set tmp to m_pHead
30:   Set pre to NULL
31:   While tmp is not NULL and tmp.getExp()  $\neq$  exp:
32:     Set pre to tmp
33:     Set tmp to tmp.getNext()
34:   Create newTerm with coeff, exp
35:   If pre is NULL:
36:     Set newTerm.setNext(m_pHead)
37:     Set m_pHead to newTerm
38:   Else If tmp is NULL:
39:     Set pre.setNext(newTerm)
40:   Else If tmp.getExp() == newTerm.getExp():
41:     Set tmp.setCoe(tmp.getCoe() + newTerm.getCoe())
42:   Else:
43:     Set pre.setNext(newTerm)
44:     Set newTerm.setNext(tmp)
45: Method printPolynomial()
46:   Set p to m_pHead
47:   While p is not NULL:
48:     If p.getCoe() == 1:
49:       If p.getExp() == 0:
50:         Print p.getCoe()
51:       Else:
52:         Print "x" + p.getExp()
53:       Print "-"
54:     If p.getExp() == 0:
55:       Print "1"
56:     Else:
57:       Print "x" + p.getExp()
58:       If p.getExp() == 0:
59:         Print p.getCoe()
```

Method: calculate(x)

---

**Algorithm 1** calculate(x)

---

```
1: Initialize result to 0
2: Set p to m_pHead
3: while p is not NULL do
4:   Initialize tmp to 1
5:   for i from 0 to p.getExp() - 1 do
6:     tmp ← tmp * x
7:   end for
8:   result ← result + p.getCoe() * tmp
9:   p ← p.getNext()
10: end while
11: return result
```

---

Method: differentiation()

---

**Algorithm 2** differentiation()

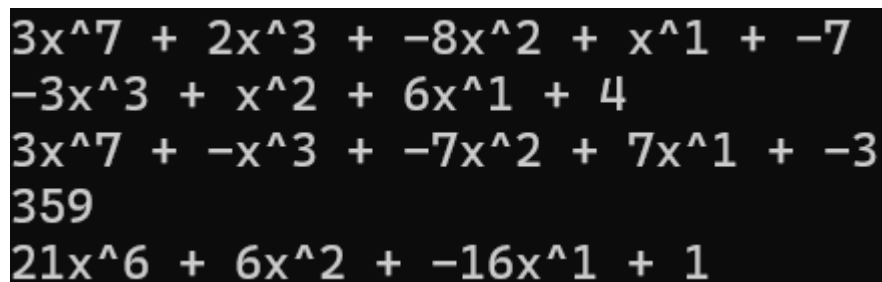
---

```
1: Create Polynomial result
2: Set p to m_pHead
3: while p is not NULL do
4:   if p.getExp() ≠ 0 then
5:     Call result.addTerm with (p.getCoe() * p.getExp(), p.getExp() - 1)
6:   end if
7:   p ← p.getNext()
8: end while
9: return result
```

---

main은 예시이므로 포함하지 않았다.

## 결과 화면



```
3x^7 + 2x^3 + -8x^2 + x^1 + -7
-3x^3 + x^2 + 6x^1 + 4
3x^7 + -x^3 + -7x^2 + 7x^1 + -3
359
21x^6 + 6x^2 + -16x^1 + 1
```

위의 두 출력은 각각의 다항식이다.

이후 출력은 두 다항식을 더한 결과이다.

아래 출력은 더한 다항식에 2를 넣은 결과이다.

마지막 줄은 더한 다항식을 미분한 결과이다.

## 고찰

다항식의 각 항이 계수와 차수를 가지는 텀인데, 이것을 연산자 오버로딩 할 때 문제가 생겼다. 문제를 해결하기 위해 찾아본 결과, 얇은 복사, 깊은 복사의 차이에 대한 문제였다. 그저 + 오버로딩만으로는 클래스 덧셈 연산자를 정의하면 main에서 덧셈을 실행할 때, 대입 이후 소멸자가 호출되어 주소값에 할당된 클래스가 소멸한다. 따라서, 메모리에 독립적인 깊은 복사를 이용하여 연산자를 오버로딩하였다. 복사 생성자와 대입 연산자 오버로딩을 통해 깊은 복사를 구현하였고, 이를 통해 덧셈, 뺄셈 연산자가 잘 작동되도록 만들었다.

6.

## 문제 설명

6번 문제는 전통 놀이인 무궁화 꽃이 피었습니다를 구현하는 문제이다.

술래가 뒤를 보면 움직이면 안 되고, 술래가 앞을 볼 때 플레이어가 움직여 20번 움직이면 성공, 뒤를 볼 때 걸리면 패배하는 규칙이다.

술래가 앞을 볼 때, 단어를 1~5개를 출력하여 2~10초 동안 앞을 보는 상태가 지속된다. 뒤를 돌아보면 바로 빨간 불이 되고 매초 2~6개씩 글자가 출력되다가 !를 출력하면 앞으로 다시 돌아본다.

플레이어는 화살표 오른쪽 키를 입력하여 앞으로 이동한다.

## 알고리즘

---

**Algorithm 1** Red Light, Green Light Game

---

```
1: Initialize flag as false
2: Define class TaggerState with virtual function toward
3: Define class Back inheriting from TaggerState
4:   Function toward sets flag to false
5: Define class Front inheriting from TaggerState
6:   Function toward sets flag to true
7: Define class Player with attributes state and progress
8:   Constructor initializes state to NULL and progress to 0
9:   Destructor deletes state if not NULL
10:  Function SetState:
11:    Initialize static variables tmp_state, cur_state, tmp, randn, idx
12:    Get current time and extract seconds
13:    Set cursor position to (0,0)
14:    If tmp_state is true:
15:      Print "Red Light!"
16:    If cur_state is false:
17:      Delete current state and assign new Front state
18:      Set cur_state to true
19:      If randn is -1, generate random number between 2 and 10
20:      If seconds have changed:
21:        If randn is 0:
22:          Set randn to -1, tmp_state to false, cur_state to false
23:          Clear the line
24:        Else:
25:          Decrement randn
26:      Else:
27:        If cur_state is false:
28:          Delete current state and assign new Back state
29:          Set cur_state to true
30:          Print "Green Light!" with cursor movements
31:          Generate random number between 2 and 5
32:          If seconds have changed:
33:            Print part of "Green Light!" based on random number
34:            If fully printed:
35:              Set randn to -1, idx to 0, cur_state to false, tmp_state to
true
36:      Update tmp with current seconds
37:    Function toward:
38:      Call state's toward function
39:      Increment progress
40:    Function GetProgress returns progress
41:    Function GetState returns state
42:  Main function:
43:    Instantiate Player object
44:    Print game instructions
45:    Wait for key press
46:    While true:
47:      If player progress is 20:
48:        Print "Win!!" and exit
49:      If flag is true:
50:        Print "lose.." and exit
51:      Call player's SetState
52:      If key pressed:
53:        Read key, print "Forward."
54:        Call player's toward
55:      Print current progress with cursor movements
```

---

## 결과 화면

```
Green Lig

start | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ | end ( 20/20 )
Win!!
exit the program
```

20번을 다 눌러 게임에서 승리했다.

```
Red Light!

start | @@@@@@@@@@@@@@----- | end ( 6/20 )

lose..
exit the program
```

이것은 걸려서 패배한 것이다.

## 고찰

6번에서 window와 console을 사용할 줄 몰라 여러 부분을 찾아보면서 공부하였다. 커서의 위치를 바꾸는 것과 출력을 조절하는데, 이전의 값이 남아있는 문제가 계속 발생하였다. 이 부분을 수정하는 작업이 오래 걸린 부분이었다. 그리고 virtual function을 처음 배워 어떻게 해야 할지 몰랐지만, 오버라이딩을 그냥 하면 된다는 것을 알고 이해하였다.



7.

### 문제 설명

7번 문제는 템플릿을 사용하여 스택을 구현하는 프로그램이다.

메서드는 push, pop, top, isEmpty, print, exit를 구현해야 한다.

push는 데이터를 스택에 저장한다.

pop은 데이터를 스택에서 빼온다.

top은 스택에 쌓인 데이터 중 가장 위(참조할 수 있는 데이터)를 반환한다.

isEmpty는 스택이 비었는지 확인한다.

print는 스택을 순서대로 출력한다.

exit은 프로그램을 종료한다.

### 알고리즘

```

Class Node<T>
Private:
    m_Next: Node<T>
    m_Data: T
Public:
    Function Constructor(data: T)
        m_Data = data
        m_Next = null
    Function Destructor()
        m_Data = 0
        m_Next = null
    Function getData() : T
        Return m_Data
    Function getnext() : Node<T>
        Return m_Next
    Function setData(data: T)
        m_Data = data
    Function setNext(next: Node<T>)
        m_Next = next

```

```

Class Stack<T>
Private:
    m_Top: Node<T>
Public:
    Function Constructor()
        m_Top = null
    Function Destructor()
        While m_Top != null
            temp = m_Top
            m_Top = m_Top.getnext()
            Delete temp
    Function push(data: T)
        newNode = New Node<T>(data)
        newNode.setNext(m_Top)
        m_Top = newNode
    Function pop() : T
        If m_Top == null
            Print "Stack is empty"
            Return -1
        temp = m_Top
        data = m_Top.getData()
        m_Top = m_Top.getnext()
        Delete temp
        Return data
    Function isEmpty() : Boolean
        Return m_Top == null
    Function top() : T
        If m_Top == null
            Print "Stack is empty"
            Return -1
        Return m_Top.getData()
    Function print()
        temp = m_Top
        While temp != null
            Print temp.getData()
            temp = temp.getnext()
        Print newline

```

## 결과 화면

```
0
stack print:c b a
c
stack print:b a
b
stack print:b a
b
stack print:a
a
stack print:
1
```

a,b,c를 차례차례 push했다.

이후 ismepty를 호출했을 때 0이 출력된다.

print 커맨드를 출력하면 넣은 순서의 반대로 c b a가 출력된다.

pop한 후 print 커맨드를 출력하면 b a가 출력된다.

top을 하여 b가 출력됐다.

다시 print를 출력하면 b a가 출력된다.

pop을 하고 b가 출력되고, print하면 a가 출력된다.

pop을 한 번 더 하면 a가 출력된다.

print하면 아무것도 출력되지 않는다.

ismepty를 하면 스택이 비어있으므로 1이 반환되어 출력된다.

## 고찰

스택은 이전 과제에서 구현하여 사용하였기에 템플릿 기능만 추가하여 구현하였다. 하지만, 템플릿에 익숙하지 않아 자료형을 지정하는 것에서 오류가 많이 생겼다. 해당 부분에 대해 공부하였고, 이후 과제에 적용해야겠다고 생각했다.

8.

## 문제 설명

8번은 템플릿을 사용하여 큐를 구현하는 프로그램이다.

메서드는 enqueue, dequeue, front, isEmpty, print, exit를 구현해야 한다.

enqueue는 데이터를 큐에 저장한다.

dequeue는 데이터를 큐에서 빼온다.

front는 큐에 저장된 데이터 중 가장 앞(참조할 수 있는 데이터)를 반환한다.

isEmpty는 큐가 비었는지 확인한다.

print는 스택을 순서대로 출력한다.

exit은 프로그램을 종료한다.

알고리즘

Node Class

Algorithm 1 Node Class

**Class** Node

**Attributes:**

*m\_Next*: Pointer to the next node  
*m\_Data*: Data stored in the node

**Method** Node(*data*: *T*):

Create a new node with the given data  
*m\_Data* ← *data*  
*m\_Next* ← **null**

**Method** ~Node():

Destructor for the node  
Free memory

**Method** getData(): *T*

Return *m\_Data*

**Method** getNext(): Node

Return *m\_Next*

**Method** setData(*data*: *T*):

Set *m\_Data* to *data*

**Method** setNext(*next*: Node):

Set *m\_Next* to *next*

Algorithm 2 Queue Class

**Class** Queue

**Attributes:**

*m\_Front*: Pointer to the front node of the queue  
*m\_Back*: Pointer to the back node of the queue

**Method** Queue():

Create an empty queue  
*m\_Front* ← **null**  
*m\_Back* ← **null**

**Method** ~Queue():

Destructor for the queue  
Free memory

**Method** enqueue(*data*: *T*):

Create a new node with the given data  
If *m\_Front* is **null**:  
Set *m\_Front* and *m\_Back* to the new node  
Else:  
Set the next node of *m\_Back* to the new node  
Set *m\_Back* to the new node

**Method** dequeue(): *T*

If *m\_Front* is **null**:  
Print "Queue is empty"  
Return **null**  
Create a temporary node and set it to *m\_Front*  
Get the data from the temporary node  
Set *m\_Front* to the next node of the temporary node  
If *m\_Front* is **null**:  
Set *m\_Back* to **null**  
Free memory for the temporary node  
Return the data

**Method** isEmpty(): boolean

Return true if *m\_Front* is **null**, false otherwise

**Method** front(): *T*

Return the data from *m\_Front*

**Method** print():

Create a temporary node and set it to *m\_Front*  
While the temporary node is not **null**:  
Print the data from the temporary node  
Set the temporary node to the next node of the temporary node

## 결과 화면

```
1
1 2
1 2 3
1 2 3 4
2 3 4
3 4
4
1
```

1234를 차례차례 입력하고 하나씩 dequeue하여 출력하였다.  
마지막은 isEmpty를 호출하면 1이 나온다.

## 고찰

스택과 비슷한 구현이다. 첫 노드와 마지막 노드를 참조할 수 있는 포인터를 두어 양 끝에서 노드의 삽입, 삭제 역할을 맡게 된다.

다른 기능은 스택과 같기 때문에 구현이 비슷했다.

9.

## 문제 설명

9번은 템플릿을 사용하여 이진 탐색 트리를 구현하는 프로그램이다.

자연수 10~99를 입력하여 출력한다.

출력 시엔 각 행에 하나의 노드 값만 출력되게 해야 한다.

노드의 왼쪽, 오른쪽 자식 노드 중 하나가 비어있다면 있는 것처럼 공백을 출력해야 한다. 노드의 왼쪽과 오른쪽의 자식 노드들은 같은 거리를 가지게 출력된다.

## 알고리즘

**Algorithm 1** Node class

---

```

1: procedure NODE
2:   Node[Ti]* m_left
3:   Node[Ti]* m_right
4:   T m_data
5:   int depth
6: end procedure
7: function NODE
8:   m_data ← 0
9:   depth ← 0
10:  m_left ← NULL
11:  m_right ← NULL
12: end function
13: function ~NODE
14:   m_data ← 0
15:   depth ← 0
16:   m_left ← NULL
17:   m_right ← NULL
18: end function
19: function SETLEFT(left: Node[Ti]* )
20:   m_left ← left
21: end function
22: function SETRIGHT(right: Node[Ti]* )
23:   m_right ← right
24: end function
25: function SETDATA(data: T)
26:   m_data ← data
27: end function
28: function SETDEPTH(depth: int)
29:   this→depth ← depth
30: end function
31: function GETLEFT
32:   return m_left
33: end function
34: function GETRIGHT
35:   return m_right
36: end function
37: function GETDATA
38:   return m_data
39: end function
40: function GETDEPTH
41:   return depth
42: end function

```

---

**Algorithm 1** BST

---

```

1: function BST
2:   m_root ← NULL
3: end function
4: function ~BST
5:   deleteTree(m_root)
6: end function
7: function SETROOT(root: Node[Ti]* )
8:   m_root ← root
9: end function
10: function GETROOT
11:   return m_root
12: end function
13: function BUILD(arr: T[], start: int, end: int)
14:   if start  $\neq$  end then
15:     return
16:   end if
17:   mid ← (start + end) / 2
18:   insert(arr[mid], 0)
19:   build(arr, start, mid - 1)
20:   build(arr, mid + 1, end)
21: end function
22: function INSERT(n: T, depth: int)
23:   newNode ← createNode(n)
24:   if m_root = NULL then
25:     m_root ← newNode
26:     return
27:   end if
28:   cur ← m_root
29:   while True do
30:     if n < cur→getData() then
31:       if cur→getLeft() = NULL then
32:         newNode→setDepth(depth + 1)
33:         cur→setLeft(newNode)
34:         return
35:       else
36:         cur ← cur→getLeft()
37:         newNode→setDepth(depth + 1)
38:       end if
39:     else
40:       if cur→getRight() = NULL then
41:         newNode→setDepth(depth + 1)
42:         cur→setRight(newNode)
43:         return
44:       else
45:         cur ← cur→getRight()
46:         newNode→setDepth(depth + 1)
47:       end if
48:     end if
49:   end while
50: end function

```

---

main은 임의로 작성한 예시이므로 첨부하지 않았다.



## 결과 화면

```
Microsoft Visual Studio 디버그 × + ▾  
input array = 12 21 37 46 59 65 74 83 89 94  
94  
89  
83 74  
65  
59 46  
37  
21  
12  
C:\Users\hojun\OneDrive\바탕 화면\개발 폴더\  
xe(프로세스 36752개)이(가) 종료되었습니다(코  
디버깅이 중지될 때 콘솔을 자동으로 닫으려면
```

위는 main 함수의 내용을 출력 예시와 같이 만들었을 때 나오는 결과이다.  
출력 예시와 같이 나온다.

## 고찰

bst의 메모리를 해제하는 부분에서 굉장히 여러 고민을 하였다.

소멸자를 재귀로 호출하는 것을 생각해본 적이 없었기 때문에 실행이 될까했지만 평소 class 안의 내용을 .으로 참조할 때 소멸자가 리스트에 있는 것을 보고 가능할까 생각을 했다. 결국 소멸자만으로 구현을 하였고, 메모리 누수가 나는 것을 방지할 수 있었다. 이것으로 class 내부에서 메모리 관리를 할 수 있었다. 또한 출력 시 자식 노드의 거리가 같아야 한다는 조건에서 어떤 출력을 해야하는 지 고민했는데, 오른쪽 자식 노드는 상관없고, 왼쪽 자식노드의 오른쪽 자식 노드를 보고 개행을 해야할지 말아야 할지를 판단의 지표로 삼았다. 이 조건을 통해 제안서의 예시와 같은 출력을 할 수 있었다.

하지만, 조건의 이해가 어려워 구현보다 문제의 해석, 문제 조건 등을 분석하고 질문하는 것이 더 힘들었던 문제이다.