객체지향 프로그래밍 3 차 과제

2018202046 이준휘

1. Assignment 1

- 문제 설명

  해당 문제는 끝말잇기를 Linked List 를 통해 구현하는 문제이다. 만약 단어가 연결되지 않거나 중복이면 특정 문장을 출력한다

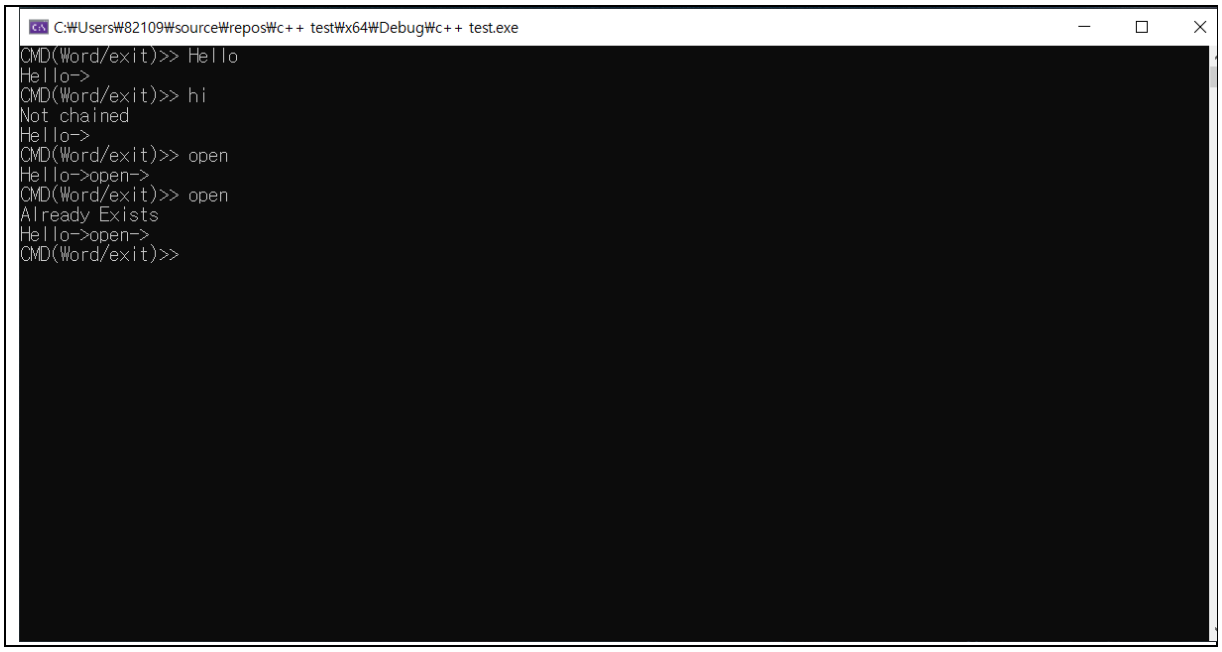
- 해결 과정

  데이터를 저장하는 class 에는 다음 주소를 가르키는 WORD* Link 와 저장된 단어인 char words[1024]가 존재한다. void Addlist(WORD** head, char* input) 함수는 LinkedList 에 단어를 추가하는 함수이다. 데이터를 새로 동적할당한 WORD* temp 에 넣고 만약 처음 입력한 단어일 경우 head 를 해당 값으로 바꾼다. 그렇지 않으면 반복문을 통해 마지막 위치 전까지 이동하는데 만약 이 사이에 같은 단어를 strcmp()로 판별할 경우 문장을 출력하고 동적할당을 해제한 뒤 함수를 종료한다. 만약 같은 단어를 찾지 못하고 마지막 직전으로 온 경우 마지막 문자를 알려주는 함수 char EndWord(char*)를 통해 이어지지 않을 때, 문장을 출력하고 동적할당을 해제한 뒤 함수를 종료한다. 위의 경우가 아닌 경우 Linked List 마지막에 해당 노드를 추가한다.

  메인 함수에서 WORD* head 를 선언하고 명령을 받는다. 만약 명령이 exit 일 경우에는 기존 동적할당이 되어있는 LinkedList 를 할당 해제 후 함수를 종료한다. 만약 아닐 경우에는 Addlist()를 통해 해당 단어를 추가한 뒤 Printlist()를 통해 단어들을 출력한다.

- 결과 화면

결과



- 고찰

  해당 문제를 푸는 과정에는 크게 어려움은 없었다. 하지만 이 문제를 통하여 1 차원 Linked List 가 무엇인지 알게 되었고, 어떻게 활용할 수 있는지 알게 된 과제였다. 또한 외부 함수에서 호출할 때 해당 클래스에 값을 추가하고 싶으면 더블 포인터를 사용해야 한다는 것을 새롭게 알게되었다.

2. Assignment 2

- 문제 설명

  해당 문제는 5 개까지만 받을 수 있는 Linked List 와 Linked List 에서 버려지는 데이터들이 모이는 이진 트리를 구현하는 것이다. Enqueue 를 통해 숫자를 추가하고, Dequeue 를 통해 Queue 에 있던 데이터를 이진 트리로 옮긴다. Print_queue 는 Linked List 를 출력하고, SEARCH 는 해당 데이터를 BST 에서 탐색한다. PRINT 는 pre-order, in-order, post-order 방식으로 각각 구현한다. EXIT 을 받을 시에는 프로그램을 종료한다. 입력이 잘못된 경우에는 해당하는 오류코드를 출력한다.

- 해결 과정

  BSTNode class 에는 값을 담을 int data 와 BSTNode* left, right 를 통해 이진트리가 짜여있다. 그리고 메소드 함수 PrintPre(), PrintIn(), PrintPost()는 재귀 함수 2 개와 출력의 위치의 따라 구분된다. Search(int) 함수 또한 재귀적 용법을 사용하여 해당 위치에서 같을 경우 출력하고, 좌측 우측 포인터에 재귀 함수를 호출한다. 만약 이가 아닐 경우 오류를 출력한다. 마지막으로

Exitbst() 함수는 BSTNode*를 동적 할당 해제하는 함수이다. 이 또한 재귀적 용법으로 구현하였다.

BST class 는 BSTNode*의 시작 주소값을 가지고 있는 클래스다. InputBST(int index)를 통해 현재 위치의 값과 삽입하는 노드의 값을 비교하고 삽입하는 노드가 작을 경우 좌측으로, 클 경우 우측으로 포인터를 이동하여 비교하고, 만약 이동하는 곳이 비어있을 경우 그 자리에 삽입한다. PrintBST(int cmd)는 만약 BST 가 비어있을 경우 오류를 출력하고, cmd 의 값에 따라 PrintPre(), PrintIn(), PrintPost()를 선택하여 출력한다. SearchBST(int num)은 노드가 비어있을 경우 오류를 출력하고 아니면 Search()를 통해 값을 탐색한다. DeleteBST()는 노드가 비어있을 경우 바로 종료하고, 아닐 경우 ExitBST()를 호출한다.

Queue class 는 Linked List 가 담긴 클래스다. InsertQueue(int index)를 통해 Linked List() 노드 가장 마지막에 값을 삽입한다. Dequeue(BST&, int count) 함수는 count 에 해당하는 개수(입력된 개수를 초과할 경우 입력된 개수까지만 삭제)만큼 Linked List 를 삭제하고 이 데이터를 BST 에 입력하는 함수다. PrintQueue() Linked List 가 비어있을 경우 오류를 출력하고 아닐 경우 이를 출력하는 함수다. ExitQueue()함수는 프로그램을 종료할 때 메모리 할당을 해제하는 함수다.

Bool Command(char*)를 통해 해당 명령어가 숫자로만 구성되어있는지 확인하는 함수다.

main 함수에서 Queue que 와 BST bst 를 모두 선언한다. int count 는 Queue 의 저장된 데이터 개수를 의미한다. 한 줄의 명령을 받고 해당 명령을 ' '을 기준으로 분리하여 cmd2 에 저장한다. 만약 Enqueue 를 입력받은 경우 cmd2 가 숫자인지 확인한 후 해당 데이터를 que.InsertQueue()를 통해 삽입하고 count 를 1 늘린다. 만약 count 가 5 를 초과한 경우에는 에러를 출력한다.

만약 Dequeue 를 입력받은 경우 동일하게 cmd2 가 숫자인지를 판별하고 입력한 숫자만큼의 데이터를 que.Dequeue()를 통해 삭제 후 bst 에 집어넣는다. 그리고 count 를 알맞게 변경한다.

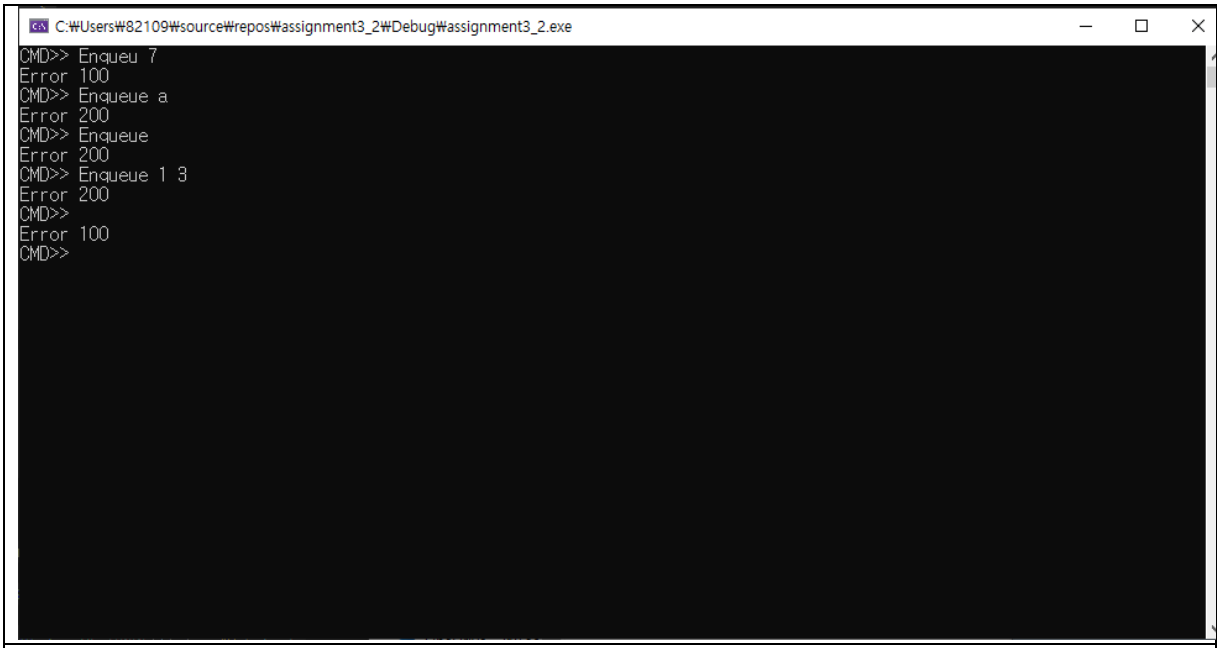
만약 Print_queue 를 입력받은 경우 함수를 통해 Linked List 를 출력한다.

PRINT 를 입력받은 경우 뒤의 단어에 따라 PRINTBST()의 값을 다르게 하여 호출한다.

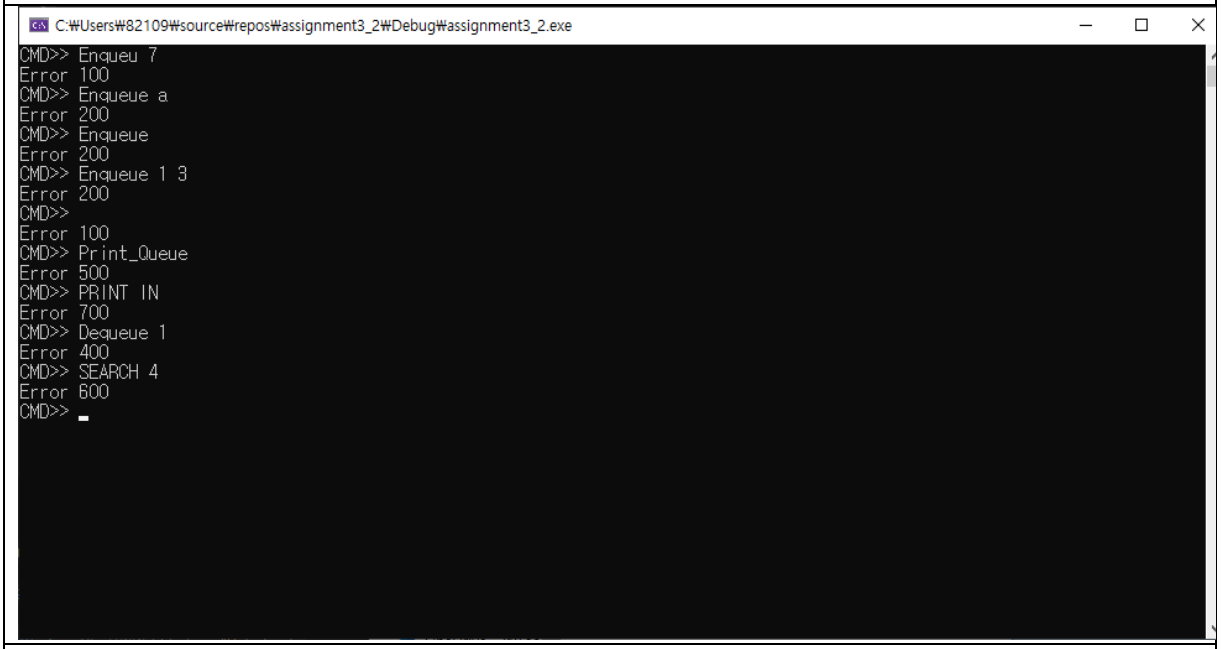
EXIT 을 입력받은 경우 que 와 bst 의 메모리를 할당 해제 후 함수를 종료한다.

- 결과 화면

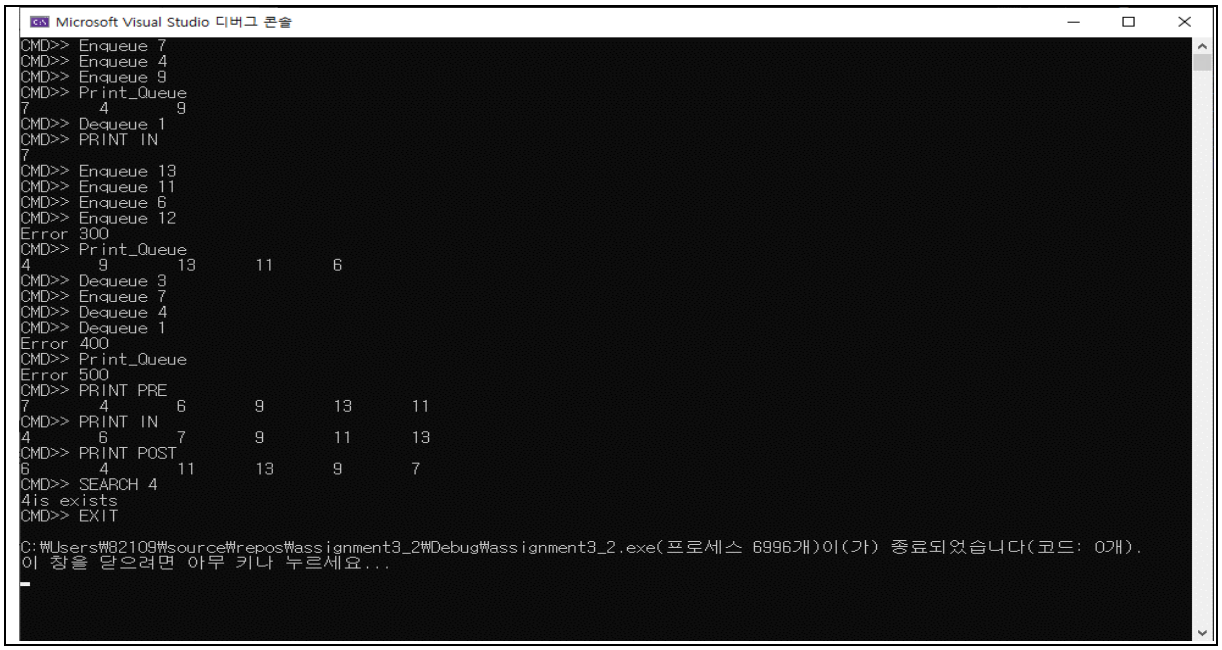
오입력


C:₩Users₩82109₩source₩repos₩assignment3_2₩Debug₩assignment3_2.exe
CMD>> Enqueu 7
Error 100
CMD>> Enqueue a
Error 200
CMD>> Enqueue
Error 200
CMD>> Enqueue 1 3
Error 200
CMD>>
Error 100
CMD>>

error


C:₩Users₩82109₩source₩repos₩assignment3_2₩Debug₩assignment3_2.exe
CMD>> Enqueu 7
Error 100
CMD>> Enqueue a
Error 200
CMD>> Enqueue
Error 200
CMD>> Enqueue 1 3
Error 200
CMD>>
Error 100
CMD>> Print_Queue
Error 500
CMD>> PRINT IN
Error 700
CMD>> Dequeue 1
Error 400
CMD>> SEARCH 4
Error 600
CMD>>

실행



- 고찰

  해당 과제를 진행하면서 이진 트리를 어떻게 사용할 수 있는지 알게되는 시간이였다. 또한 재귀적 용법을 잘 활용하면 이러한 이진 트리나 Linked List 에서 쉽고 간단하게 동작을 진행할 수 있다는 사실도 알게되었다.

3. Assignment 3

- 문제 설명

  해당 문제는 메뉴의 가격과 이름 순으로 각각 Lind List 와 BST 를 통하여 구현하고, 여기에 노드를 추가, 파일에서 불러오기, 탐색, 출력, 삭제 등의 동작을 수행하는 프로그램을 만드는 것이다.

- 해결 과정

menu_node 에 char* menu 에는 이름, int price 에는 가격이 들어간다. 그리고 menu_node* pPrev 와 pNext 를 통해 앞뒤 위치로 이동할 수 있는 양방향 Linked List 다. 만약 해당 노드를 char*과 함께 선언한 경우 해당 이름이 menu 에 들어가게 된다. 그리고 소멸자는 menu 의 메모리를 반환한다. Delete_menu_node()는 메인 함수 종료 시 모든 메모리를 재귀 함수를 통해 반환한다.

bst_node 에는 값을 가리키는 menu_node* pNode 와 bst_node* pLeft 와 pRight 로 구성되어있다.

menu_node* InputMenu(menu_node**,char*, int)함수는 데이터를 받아서 이를 Linked List 에 가격에 따라 삽입하는 역할을 한다. 동일 데이터 존재 시 오류를 출력한다. 그리고 추가한 노드의 위치를 반환한다.

void InputMenuByName(bst_node** head, menu_node* input) 함수는 추가한 노드의 위치를 받아서 이를 BST 에 추가하는 역할을 한다. 해당 입력의 이름이 현재 위치보다 먼저 나올 경우 왼쪽, 나중에 올 경우 오른쪽으로 이동한다.

PrintByName(bst_node*), PrintByPrice(menu_node*) 함수는 BST 노드를 Inorder 순으로, Linked List 를 순서대로 출력한다.

int SearchMenu(menu_node*, char*)함수는 Linked List 에서 해당 이름의 데이터가 있는지 판단하고 없을 경우 문구를 출력한다. 있을 경우 가격을 반환한다.

bool DeleteBst_node(bst_node** head, char*)함수는 입력받은 데이터를 삭제하는 함수다. 삭제를 할 경우 삭제하는 위치에 따라 삭제하는 방법이 달라진다. 기본적으로 삭제는 해당 노드를 삭제하고 해당 노드의 우측 노드는 해당 노드 좌측 노드의 우측 노드 끝에 이어 붙인다. 그리고 해당 노드의 좌측 노드를 해당 노드 자리가 있던 자리로 위치시킨다. 노드 삭제에 성공한 경우 true 를 반환하고, 삭제가 안됐을 경우 문장을 출력하고 false 를 반환한다.

DeleteMenu_node(menu_node** head, char* input)함수는 Linked List 에서 노드를 삭제하는 함수다. 삭제할 노드를 없애고 삭제한 노드 앞 뒤를 연결해준다.

ExitMenu(menu_node*), void ExitBst(bst_node*) 함수는 프로그램 종료 시 메모리를 반환하는 함수다.

메인 함수에서는 명령을 받고 해당하는 명령어에 따라 아래 동작을 수행한다.

만약 LOAD 를 입력받은 경우 menu.txt 를 불러와 ','와 '₩n'에 따라 정보를 구분하여 이를 InputMenu()를 통해 Linked List 에 입력하고, 반환된 주소값을 통해 InsputMenuByName()을 통해 bst 에 입력한다.

PRINT 를 입력받은 경우 MENU 또는 PRICE 를 추후에 입력함에 따라 bst 를 출력하거나, linked list 를 출력한다.

INSERT 를 입력받은 경우 이름과 가격을 입력받고 InputMenu()를 통해 Linked List 에 입력하고, 반환된 주소값을 통해 InsputMenuByName()을 통해 bst 에 입력한다.

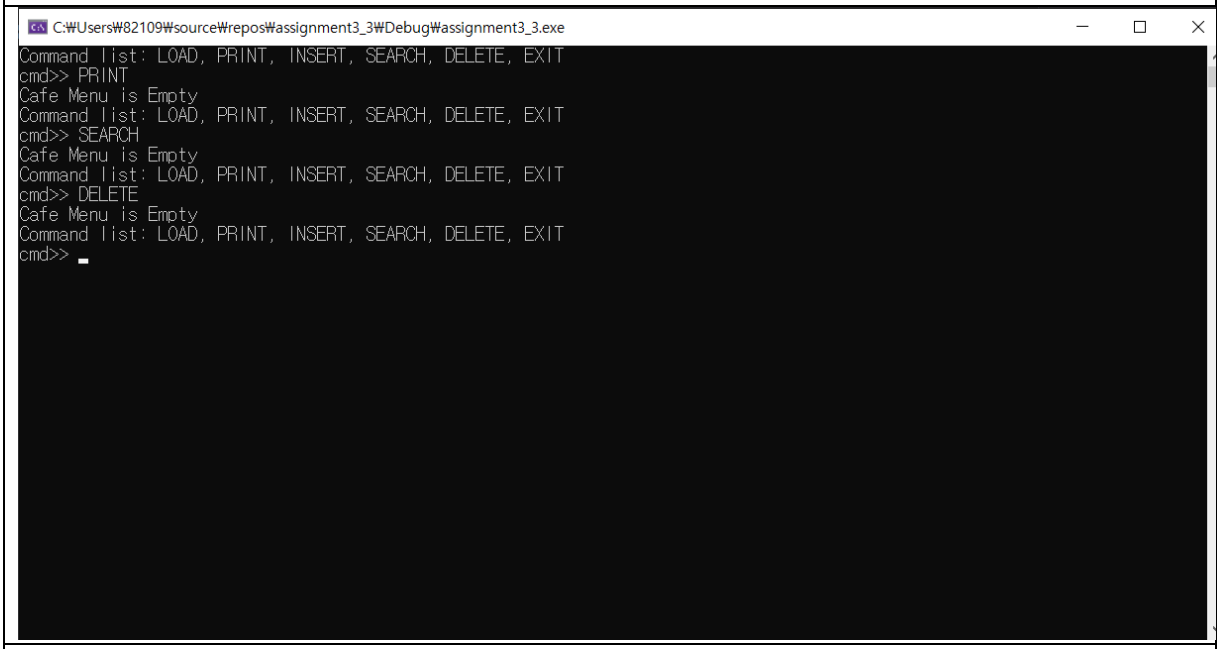
SEARCH 를 입력받은 경우 찾은 이름을 입력받고 SearchMenu() 함수를 통해 탐색한다.

DELETE 를 입력받은 경우 삭제할 이름을 입력받고 DeleteBst_node()를 통해 bst 에서 삭제되었을 경우 DeleteMenu_node()를 실행한다.

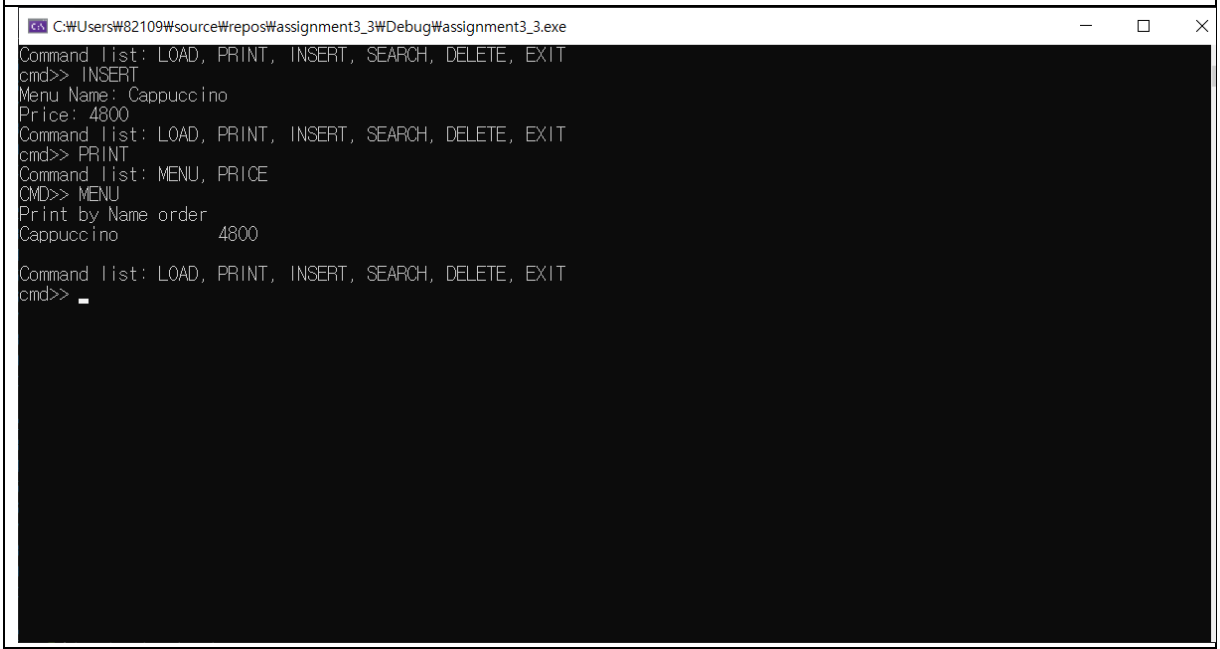
EXIT 을 입력받을 경우 bst 와 Linked List 를 동적할당 해제시킨다.

- 결과 사진

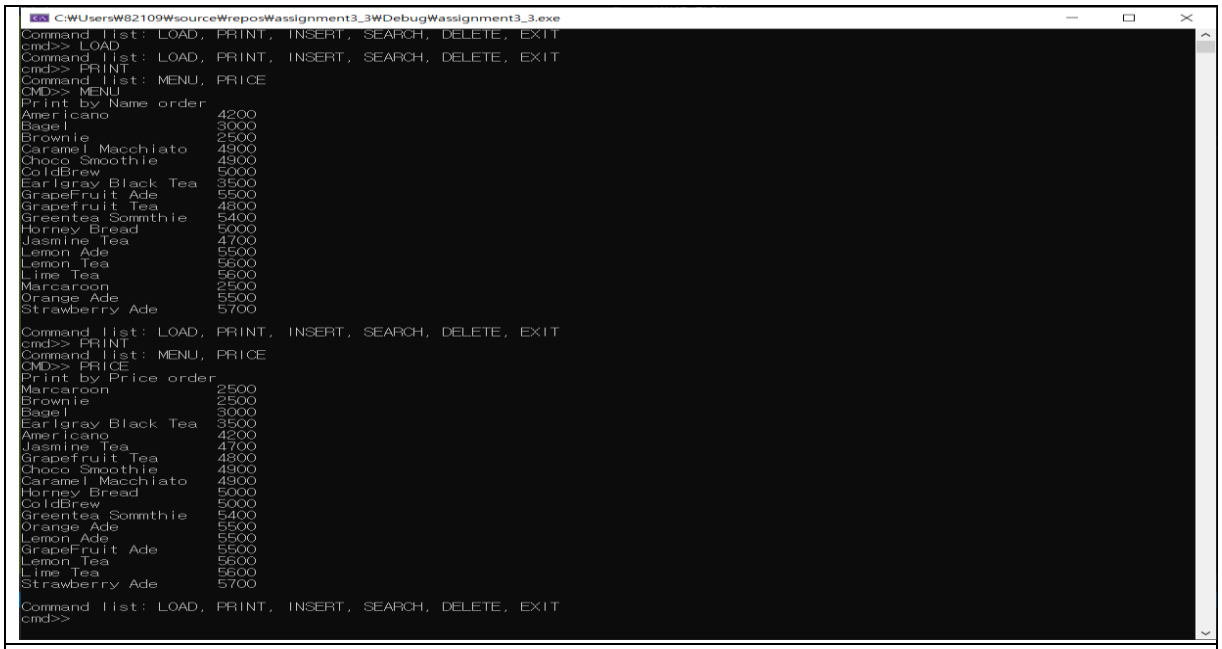
error



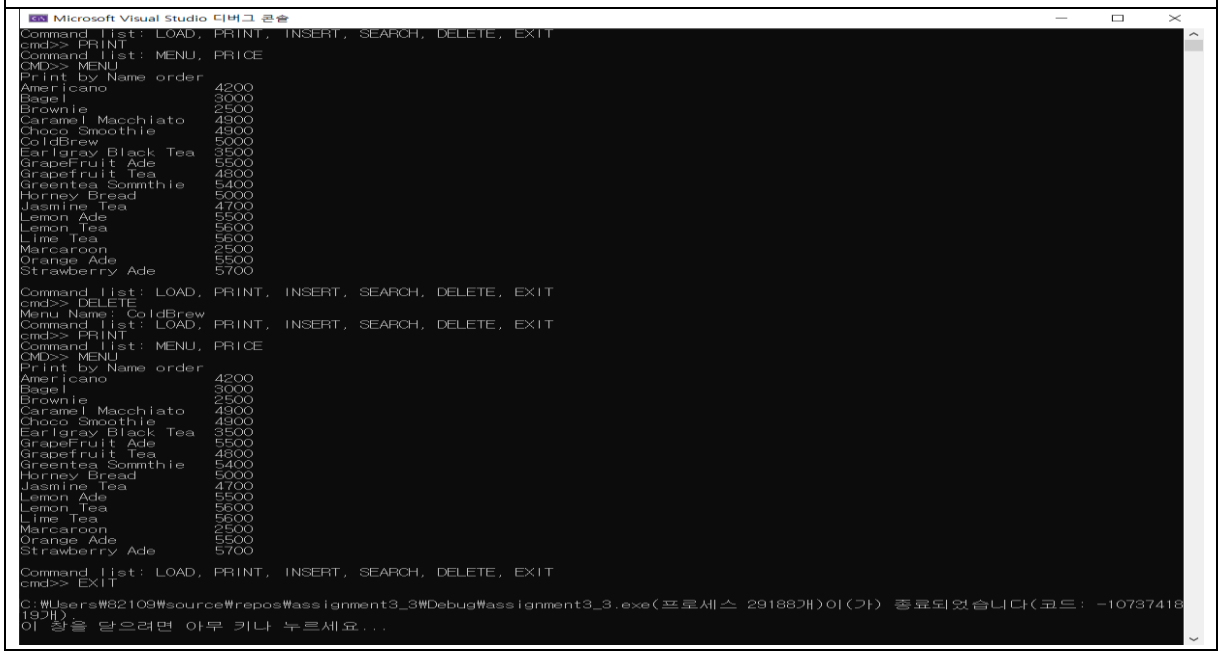
단순 입력



파일 입력



파일 삭제



- 고찰

  해당 과제는 필자가 BST 로 이루어진 데이터에서 어떻게 하면 데이터를 삽입하고 삭제할 수 있을 지 고민하고 공부하게 해주었던 과제였다. 그리고 단방향 이외의 Linked List 에 익숙해질 수 있었던 과제였다.

4. Assignment 4

- 문제 설명

  4 번 문제는 블랙잭을 구현하는 문제다. A 는 11 또는 1, J, Q, K 는 10 으로 계산된다. 각 카드는 Linked List 를 통해 저장되며 섞을 시에는 Deck 에 남아있던 카드를 모두 Discard_tray 에 올려두고 일반적인 카드 섞기 방식을 모방하여 입력한 수만큼 섞는다. 게임을 할 경우 플레이어와 딜러는 카드 2 장을 받고 딜러는 한 장을 숨기고 모두 공개한다. 플레이어는 21 이 넘지 않는 선에서 계속 받을지 말지를 선택한다. 플레이어가 멈추면 딜러는 카드를 공개하고 16~21 사이에 위치할 수 있도록 카드를 추가로 받는다. 21 을 초과하지 않는 선에서 가장 높은 값을 가진 사람이 승리한다. 카드가 13 장 미만로 남을 경우 문장을 출력하고 초기 화면으로 돌아간다. 이상으로 남을 경우 계속 할지 게임을 종료할지 선택한다.

- 해결 과정

  Deck class 에서는 카드의 문양을 담는 char data[4]와 아래 카드인 Deck* next 가 존재한다. makeCard() 함수를 통하여 52 장의 카드를 순서대로 만들어서 저장한다. Deck* CardSuf(int& count) 함수는 카드를 대중적인 방식으로 섞는 방법이다. count 로 남은 카드중에서 일부를 선택하고 아래 있던 카드를 위로 올린다. 이 때 위로 올린 카드를 재귀 함수를 사용하여 섞고 섞은 카드의 head 값을 반환함으로써 섞인 카드의 가장 윗 장의 주소를 알고 연결할 수 있다. int checkCard() 함수는 덱에 남은 카드의 개수를 확인하는 함수다. PrintCard(bool) 함수는 state 에 따라 딜러가 숨겨서 출력할 때와 모든 카드를 공개할 때 두가지 경우를 모두 출력할 수 있는 함수다. Delete Card() 카드의 메모리를 반환하는 함수다.

  Discard_tray class 는 사용한 카드가 올라가는 클래스로 Deck* head 를 값으로 갖는다. GetDiscard(Deck* input) 함수는 남은 덱을 버리는 카드 위에 올려두는 함수다. Shuffle(Deck**, int) 함수는 모든 카드를 버리는 카드 위에 올려두고 지정된 숫자만큼 셔플하는 함수다. 이때 우리는 CardSuf(52)함수를 통해 카드 52 장을 모두 셔플로 섞는다. 이를 입력받은 수만큼 반복한다. 그리고 섞은 카드를 덱에 올려둔다.

  Player 와 Dealer class 는 각 인물의 카드 Deck* card 와 카드의 최소 합인 int minsum, 그리고 A 의 개수 int Anum 을 갖는다. setCard(Deck** input) 함수는 덱에서 카드 노드 하나를 가져와 저장하고 해당되는 카드의 값에 따라 Anum 과 minsum 을 조정한다.

  메인함수에서는 우선 덱과 버리는 곳을 만든다. 그리고 게임 내용을 반복한다. 우선 버려진 카드들을 모두 불러들이고. 초기에 무작위로 섞는다. 그 후 명령을 입력받게 된다. 입력받은 명령은 ' '을 기준으로 나누어 cmd2 에 저장한다.

  game 을 입력받은 경우 플레이어와 딜러를 선언하고, 각각 2 장씩 카드를 준 후 딜러 한장을 제외하고 모두 PrintCard()를 한다. 그리고 아래 내용을 게임이 종료시까지 반복한다.

hit 입력 시 카드를 한 장 더 받고 출력을 다시 한다. 만약 이때 최소값이 21 초과 시 플레이어는 패배한다. stand 를 입력할 경우 딜러는 최소값이 16 을 넘고 또는 A 가 한 장 이상 있고 최소값이 1 을 넘으면서 19 이하일 때 카드를 getCard()를 통해 받는다. 그리고 int P1num 과 int Dealnum 을 선언하고 만약 A 를 가지고 있고 A 를 넣었을 때 21 을 초과하지 않는 경우 해당 값으로 값을 조정한다.

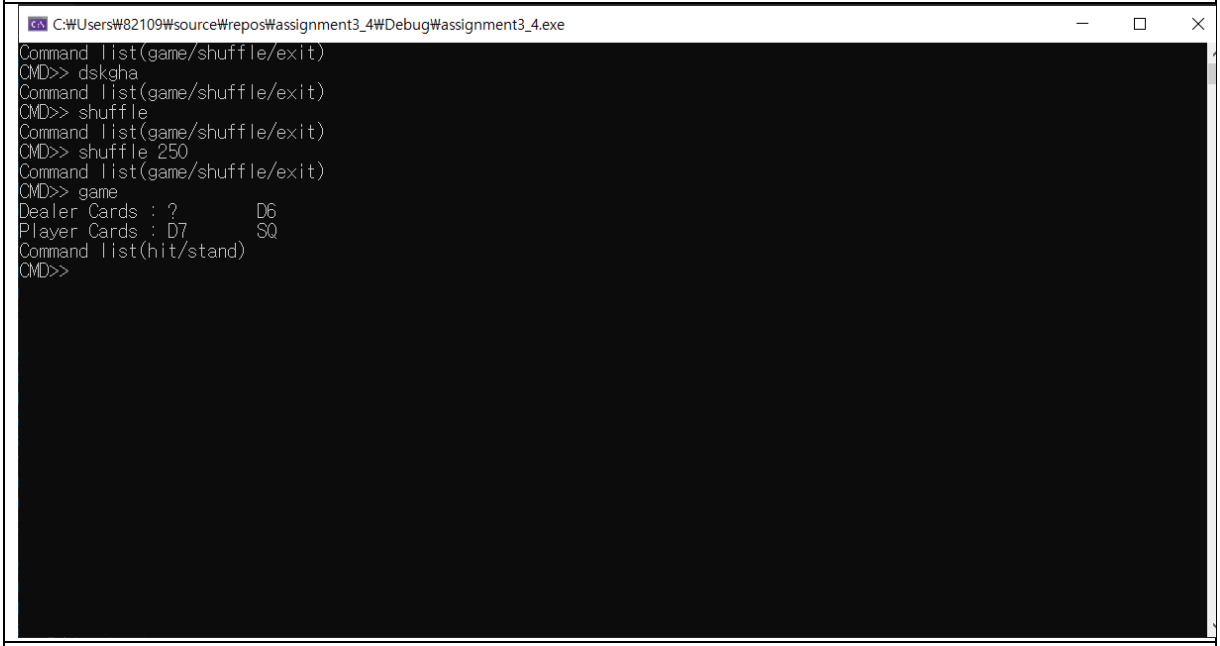
결과를 확인하여 딜러가 21 을 넘거나 딜러보다 수가 많을 시 승리하고 딜러보다 수가 작을 시 패배한다. 같을 경우는 무승부로 처리한다. 남은 카드가 13 장 미만일 경우 남은 카드를 카드는 GetDiscard()통해 Discard_tray 로 옮기고 초기로 돌아간다.

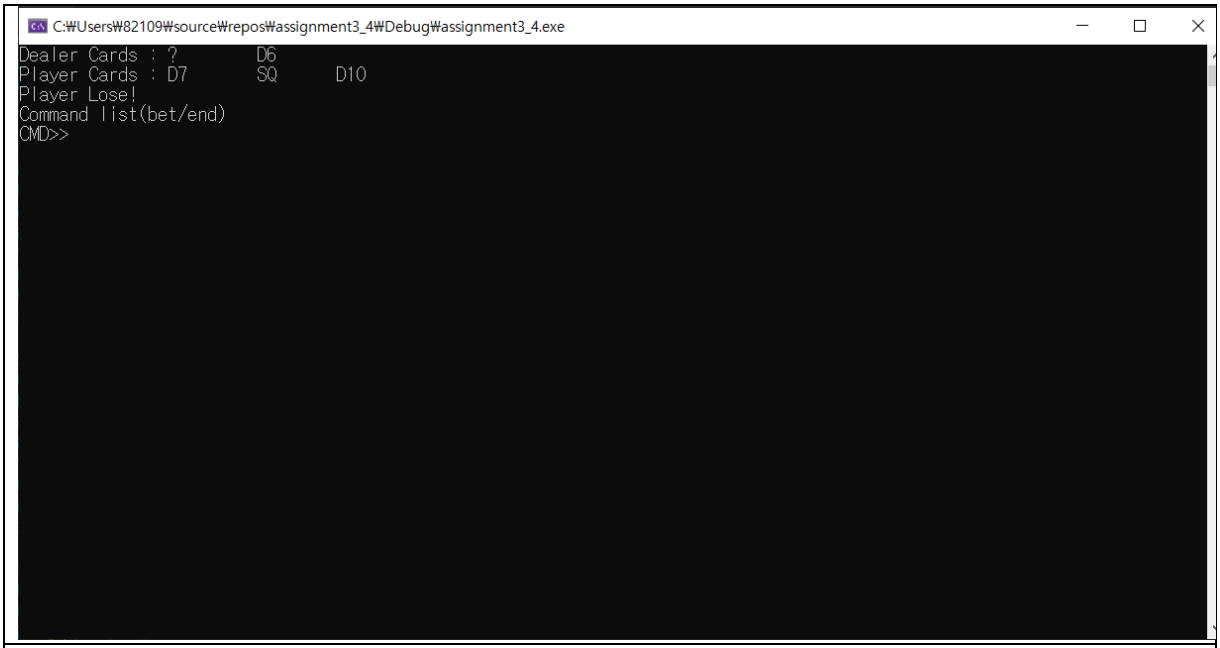
만약 13 장 이상일 경우 남은 카드로 게임을 더 진행할지 확인한다. bet 를 입력한 경우 사용했던 카드는 GetDiscard()통해 Discard_tray 로 옮긴 후 게임을 진행한다. end 를 입력한 경우 모든 카드를 BJ 로 옮긴 후 메모리 할당 해제 후 프로그램을 종료한다.

초기에 shuffle 을 입력한 경우 뒤의 문자의 숫자를 확인하여 해당 숫자를 인자로 하는 suf-Suffle()함수를 수행한다.

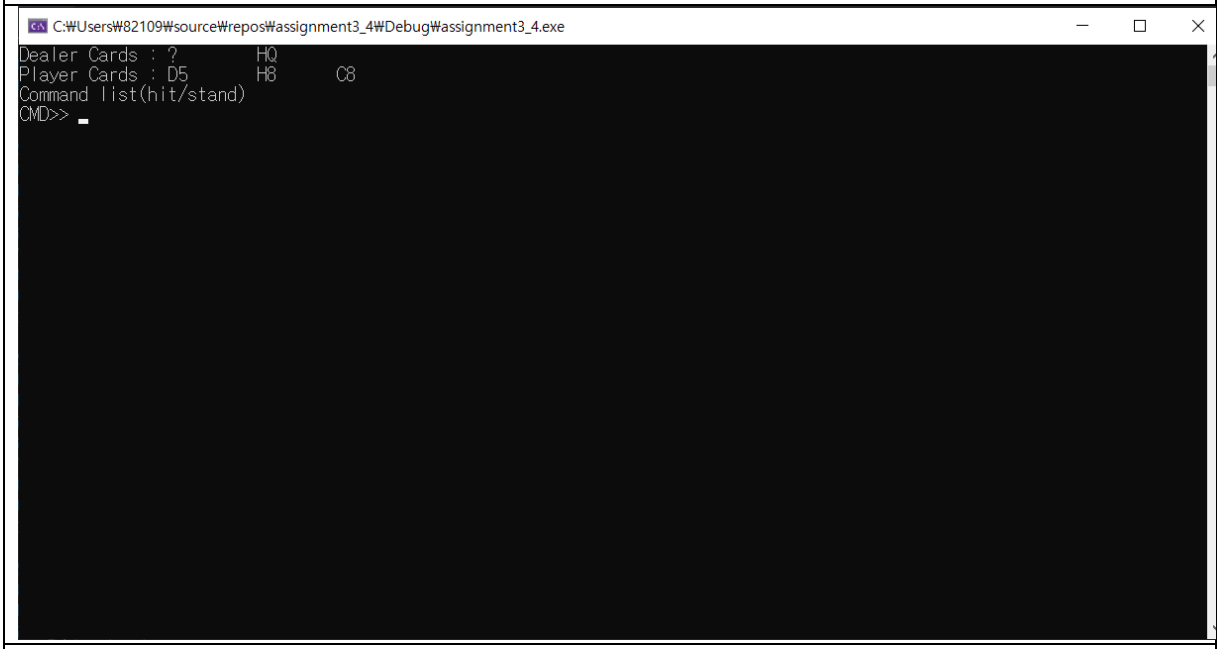
exit 을 입력한 경우 메모리 할당 해제 후 함수를 종료한다.

- 결과 사진

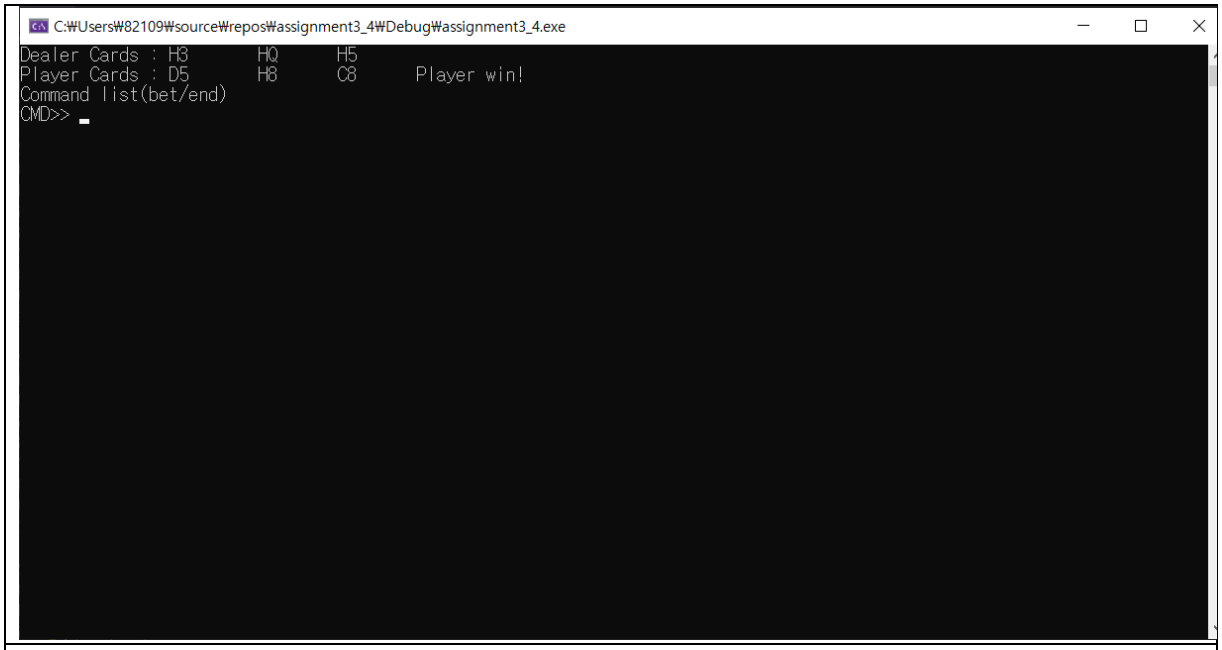
game

lose


C:₩Users₩82109₩source₩repos₩assignment3_4₩Debug₩assignment3_4.exe
Dealer Cards : ? D6
Player Cards : D7 SQ D10
Player Lose!
Command list(bet/end)
CMD>>

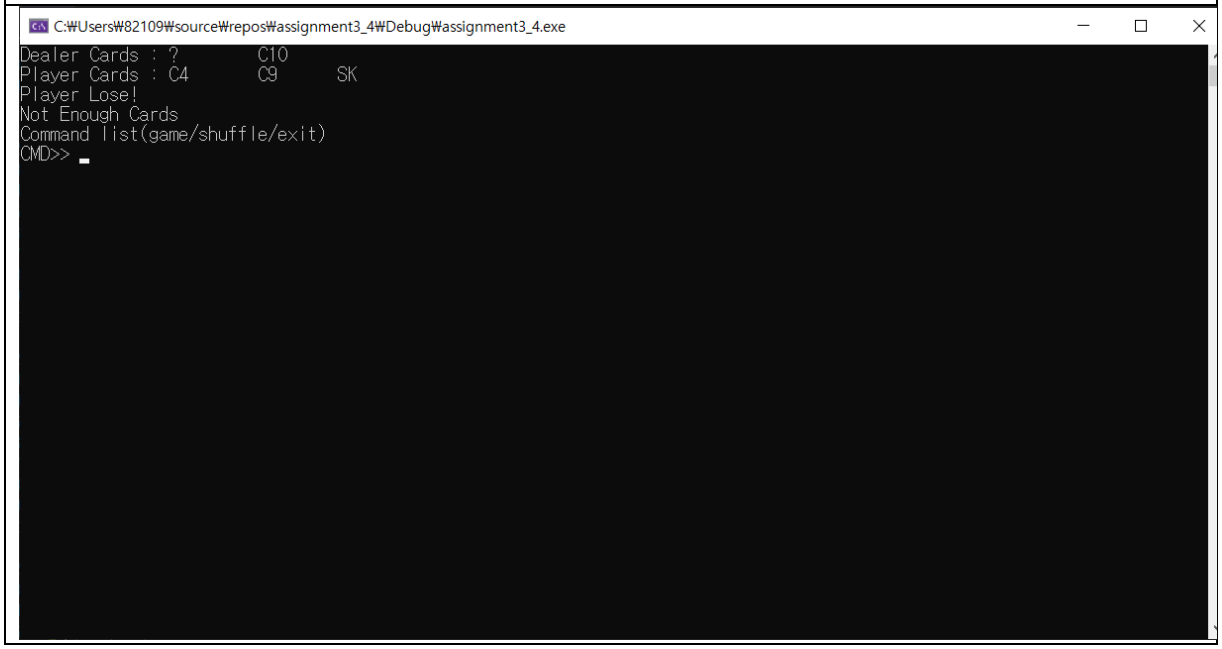
hit


C:₩Users₩82109₩source₩repos₩assignment3_4₩Debug₩assignment3_4.exe
Dealer Cards : ? HQ
Player Cards : D5 H8 C8
Command list(hit/stand)
CMD>>

win



card < 13



- 고찰

  해당 문제에를 통해 필자는 평소에 익숙했던 게임인 블랙잭을 직접 구현해보면서 흥미를 느낄 수 있었다. 셔플의 알고리즘을 고민하면서 재귀적인 용법에 좀 더 능숙해진 느낌을 받았다. 또한 규칙을 하나하나 검토해가면서 완성된 과제를 보며 보람을 느낄 수 있던 과제였다.

5. Assignment 5

- 문제 설명

해당 문제는 러시안 룰렛을 구현하는 과제다. 6 개의 노드는 원형 linked list 로 연결되어있고 rotate 를 입력할 시 무작위로 배열이 돌아간다. 그리고 shoot 을 입력한 경우 지금 가리키는 곳이 총알이 있는지를 확인하고 없을 경우 살았다는 문구를 출력한다. 그리고 배열을 한 칸 돌린다. 만약 총알이 있을 경우 죽었다는 문구를 출력하고 함수를 종료한다.

- 해결 과정

Node class 에는 총알이 들어있는지인 bool state 와 다음 노드의 주소인 node* next 가 있다. state 의 초기값은 false 로 한다.

Roulette class 에는 Node* head 를 통해 대표값을 하나 갖고 있는다. Roulette 를 선언할 때 6 개의 노드를 원형으로 연결하는데 이때 head 에 연결된 노드의 state 는 true 로 한다. 소멸자는 노드를 6 개 모두 메모리 할당 해제시킨다. rotate() 함수는 0~5 까지 수만큼 head 의 주소를 다음 위치로 바꾼다. bool shoot() 함수는 현재 head 의 state 값을 불러오고 만약 false 인 경우는 문장을 출력 후 다음 위치로 이동시키고 true 를 반환한다. state 가 true 인 경우에는 문장을 출력 후 false 를 반환한다.

메인함수에서는 Roulette russian 을 선언하고 처음 룰렛을 돌려놓는다. 그 후 아래 내용을 반복한다.

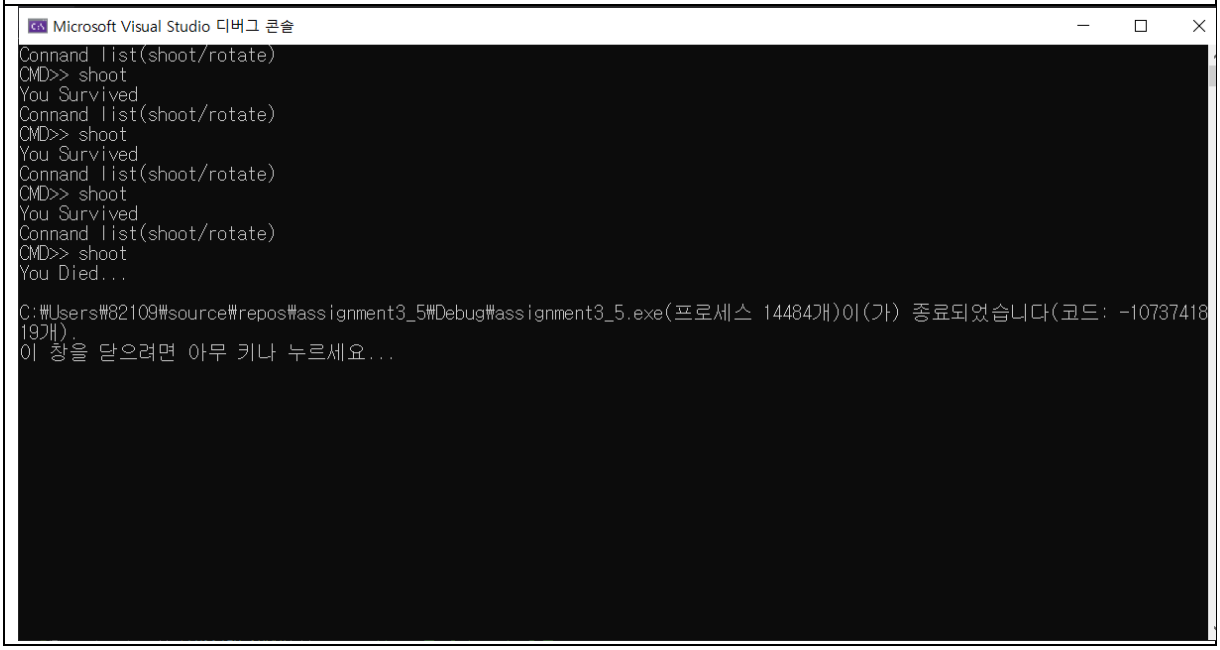
명령을 입력받고 입력받은 명령이 shoot 일 경우 russian.shoot()의 값이 false 가 될 경우 함수를 종료한다.

만약 rotate 를 입력할 경우 rotate() 함수를 통해 룰렛을 돌리고 system("cls")를 통해 화면을 지운다.

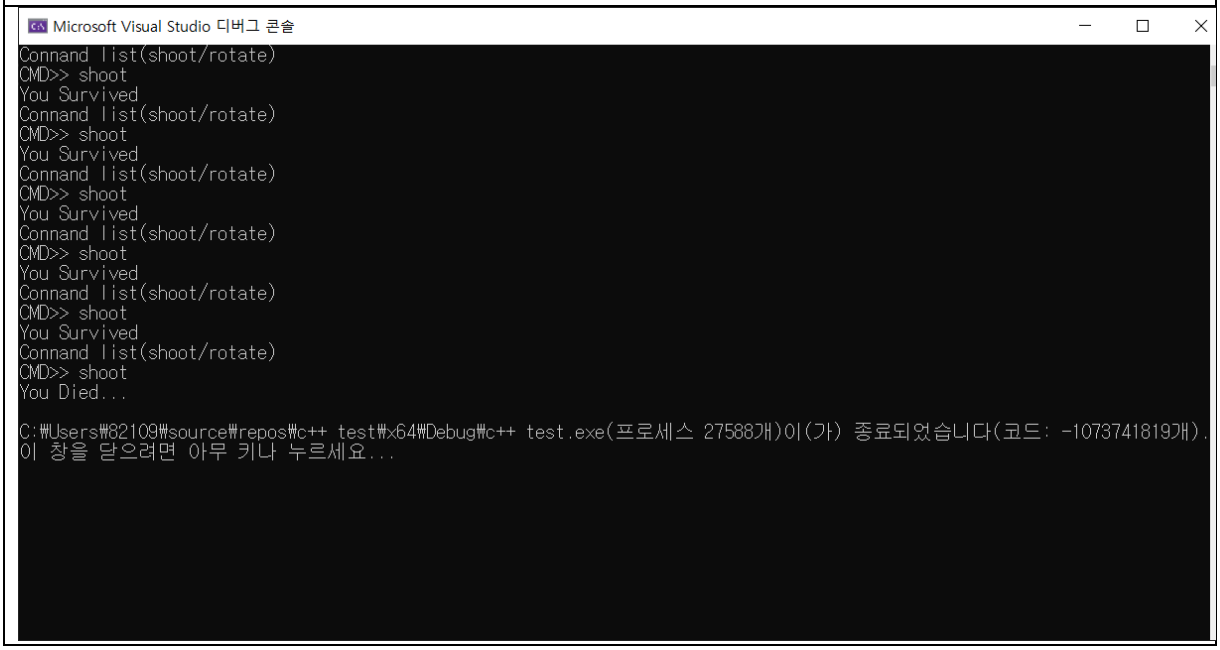
이외의 명령어는 오류로 처리한다.

- 결과 사진

game 1



game 2



- 고찰

해당 과제를 통해 원형 Linked List 를 활용할 수 있는 방법을 아는 계기가 되었다. 원형 Linked List 를 사용하면 반복되어야 하는 상황이 만들어질 때 활용하면 유용할 것 같다. 그리고 이번 과제 역시 게임을 주제로 만들었기 때문에 재미를 느끼며 과제에 임할 수 있었다.

6. Assignment 6

- 문제 설명

해당 문제는 윷놀이를 구현하는 과제다. 윳놀이를 Linked List 를 활용하여 구현한다. 게임 인원은 2 명이고 각각 4 개의 말이 있다. 모든 말이 먼저 도착지로 오면 우승이다. 윷은 60 프로 확률로 앞면이 나오며 4 개의 윷이 있다. 각 플레이어는 윷을 던지고 만약 상대를 윷이 아닌 값으로 잡거나 윷이나 모가 나온 경우에는 한 번 더 던진다. 같은 아군의 말은 업어서 한번에 이동시킬 수 있다.

- 해결 과정

해당 과제는 시간 관계상 완전히 완성하지 못했다. 해당 과제를 구현한 곳 까지만 설명한다면, 우선 Node class 를 이용하여 해당 위치의 있는 말, 사거리 교차점인지, 그리고 종료지점인지를 변수로 확인한다. void MakeMap()함수는 일반적인 길은 next 로 묶고 이전 노드는 prev 로 묶으며, 지금길은 fast 로 묶는다. voidPrintHorse()함수는 해당 위치에 어떠한 말이 있는지 출력하는 함수이다. PrintMap() 함수는 순차적으로 맵을 출력하는 함수다.

MoveLink 와 MoveNode class 는 윷이 나온 결과를 저장한다. Throw 를 통해 60 프로 확률로 앞이 나오도록 던진다. 그리고 윷이나 모일 경우에는 1 을 반환한다. UseMoveNode() 함수는 선택한 번호의 노드 윷 결과를 가져오고 해당 노드를 삭제하는 함수다.

class Player 에는 해당 플레이어의 말 위치와 도착 여부가 있다.

bool Command(int) 함수를 통해 throw 를 정상적으로 입력받을 때까지 반복한다.

MoveHorse(Node*, Node**, int) 함수는 아직 완전히 구현되지 않았지만 위치가 없었을 경우 위치를 처음에서 움직이는 거리만큼 이동시키고, 놓여있는 말은 갈림길에 있을 때 지름길로 갈 수 있도록 하는 함수다.

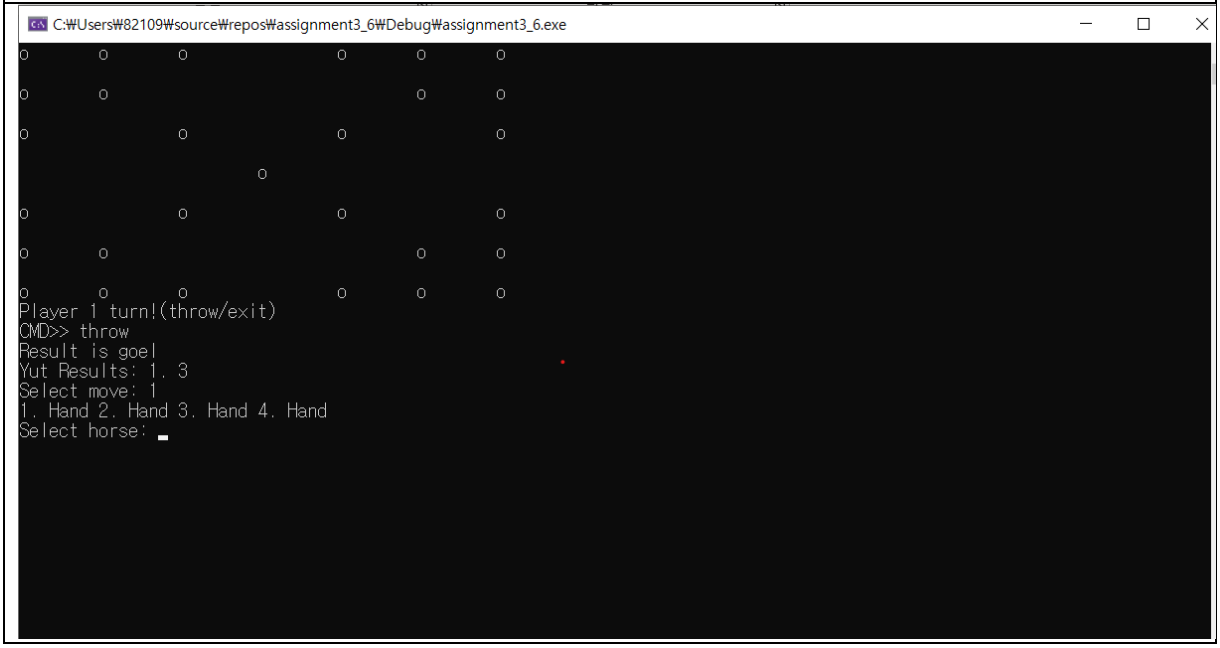
메인 함수에서 윷판과 윷, 플레이어 클래스를 선언하고 맵을 만든다. turn 을 통해 P1 과 P2 가 번갈아가면서 게임을 한다.

화면을 출력하고 Throw 를 입력받을 경우 윷이나 모가 나오면 다시 던지도록 한다. 그 후 던진 결과들을 출력하고, 어떠한 말을 이동시킬지 출력한다. UseMoveNode()를 통해 이용한 이동은 삭제하고 말을 이동시킨다.

만약 시간 분배를 잘 했더라면 그 후의 세부 조건과 완성도를 높일 수 있었던 과제인거 같아 아쉬움이 많았다.

- 결과 사진

초기



7. Assignment 7

- 문제 설명

  해당 문제는 학생 정보(이름, 학번, 전공)을 student.txt(각 데이터는 ','로 구분된다)에서 불러오고 이를 linked list 에 저장한다. 그 후 각각 정보에 따라 2 차원 Linked List 를 사용해야 한다. 학번의 경우에는 연도 순으로, 전공은 이름 순, 이름은 첫 알파벳 순으로 그룹화 되어있다. 각 그룹은 학번 순서에 따라 정렬되어있다. 그리고 정보들은 중복이 없고 2 차원 Linked List 구성 시 template 를 활용해야 한다. 그리고 허락되지 않은 입력은 없다고 가정한다. 사용자는 원하는 정렬 방식에 따라 출력을 선택한다. 선택한 출력에 맞춰서 학생 정보는 화면에 나온다.

- 해결 과정

  우선 데이터를 입력에 따라 저장할 Studenet_Node class 가 있다. 해당 함수에 인자로는 long int ID, char Major[100], name[20]이 선언되어있다. 그리고 각각의 Linked List 의 다음 노드를 가리킬 nextXXXX 가 4 개 선언되어있다. Student_Node* inputData(long, char*, char*)함수의 경우 데이터를 입력받아 nextInput Linked List 로 저장하고 저장한 노드의 주소를 반환하는 함수다. 해당 함수는 리스트의 가장 마지막에 저장해간다. Print(int mode) 함수는 mode 에 따라 각각에 맞는 next 를 사용하여 재귀적 용법을 통해 Linked List 를 출력한다. DeleteNode() 함수는 메모리 할당을 해제하는 함수다.
  각각의 정보에 따라 2 차원 Linked List 로 저장하는 class 인 IDLink, MajorLink, NameLink 가 있다. 각각의 함수에는 데이터 저장 영역인 Student_Node* head 가 존재하며 IDLink* nextRow 를 통해 2 차원까지 구현한다. PrintXXXX()함수가 존재한다. 해당 함수는 각 형식에 맞춰 Print() 함수에 인자값을 넣는다. int Compare(XXXX)함수는 분류하는 데이터의 정보에 따라 인자가 다르다. 해당 인자를 가지고 현재 head 의 데이터와 비교하여 현재 정보보다 먼저 나와야 할 경우 -1, 같을 경우 0, 나중에 나와야 할 경우 1 을 반환한다.
  InputLink(Student_Node** head, Student Node* input, int mode) 함수는 2 차원 Linked List 의 종류에 따라 next 를 다르게 하여 해당 행에 inputData()함수로 삽입된 연결하는 함수다. 어떠한 next 를 쓸 지는 mode 를 통해 결정된다. 그리고 해당 노드들끼리 ID 의 크기에 따라 정렬된다. template <T1, Tdata>를 사용하는 input2DLink(T1*, Student_Node*, Tdata, int)함수는 Compare 를 통해 현재 위치의 값과 추가하려는 값의 크기를 비교하고 순서에 따라 2 차원 Linked List 로 저장하는 함수다. T1 은 저장하려는 클래스를 의미하며, Tdata 는 분류하는 데이터, 즉 삽입하는 데이터 중 기준으로 할 정렬에 관련된 데이터를 의미한다. Student_Node*는 삽입되는 데이터를 의미하고, 마지막으로 int 는 모드에 따라 구분을 짓기 위해 사용된다.
  메인 함수에서 각각의 리스트를 모두 동적할당을 통해 선언한다. 그 후 student.txt 파일을 열고 ','와 '₩n'을 기준으로 데이터를 long inID, char* inMajor, char* inName 으로 구별하여 저장한다.

이 데이터는 우선 inputHead->inputData()를 통해 inputHead 에 저장되고 저장된 위치를 Student_Node* input 에 저장한다. 그 후 해당 데이터는 input2DLink(각각의 class, input, 판별하는 데이터, 각각의 mode)를 통해 2 차원 Linked List 에 저장된다. 그 후 1,2,3,4,5 의 입력을 받는다.

1 을 입력한 경우 1 차원 Linked List 인 inputHead 를 Print(0)를 통해 출력한다.

2 를 입력한 경우 2 차원 Linked List 인 IDHead->PrintID()를 통해 출력한다.

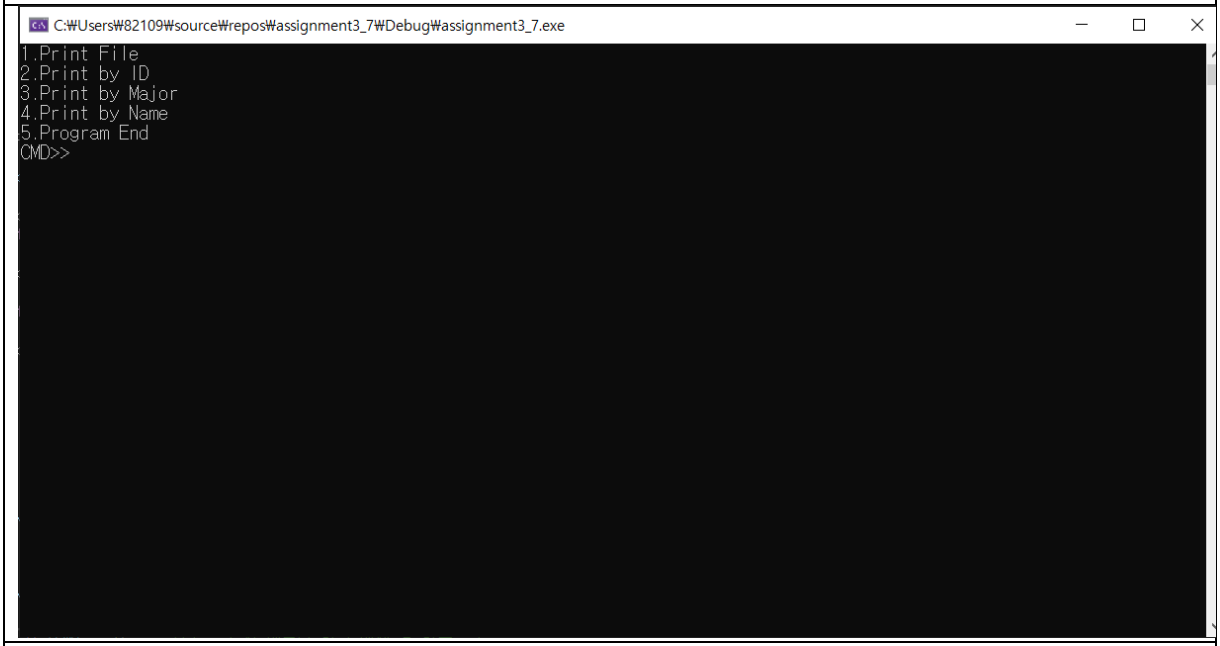
3 을 입력한 경우 2 차원 Linked List 인 MajorHead->PrintMajor()를 통해 출력한다.

4 를 입력한 경우 2 차원 Linked List 인 NameHead->PrintName()을 통해 출력한다.

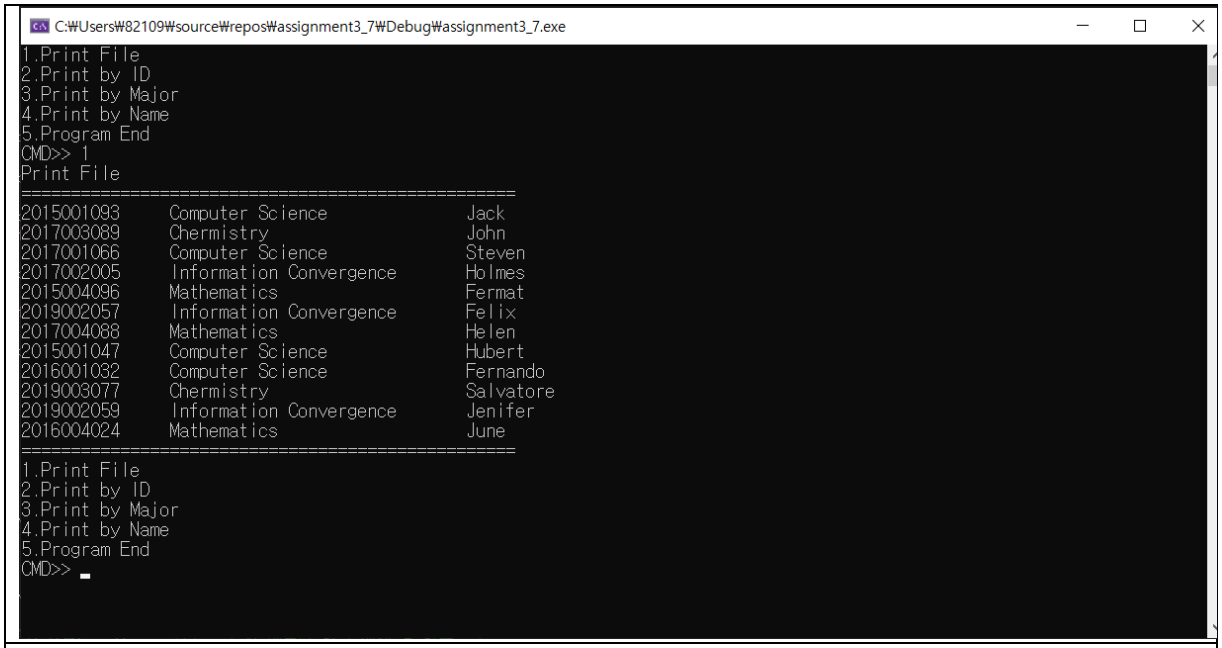
5 를 입력한 경우 메모리 할당을 해제 후 함수를 종료한다.

- 결과 사진

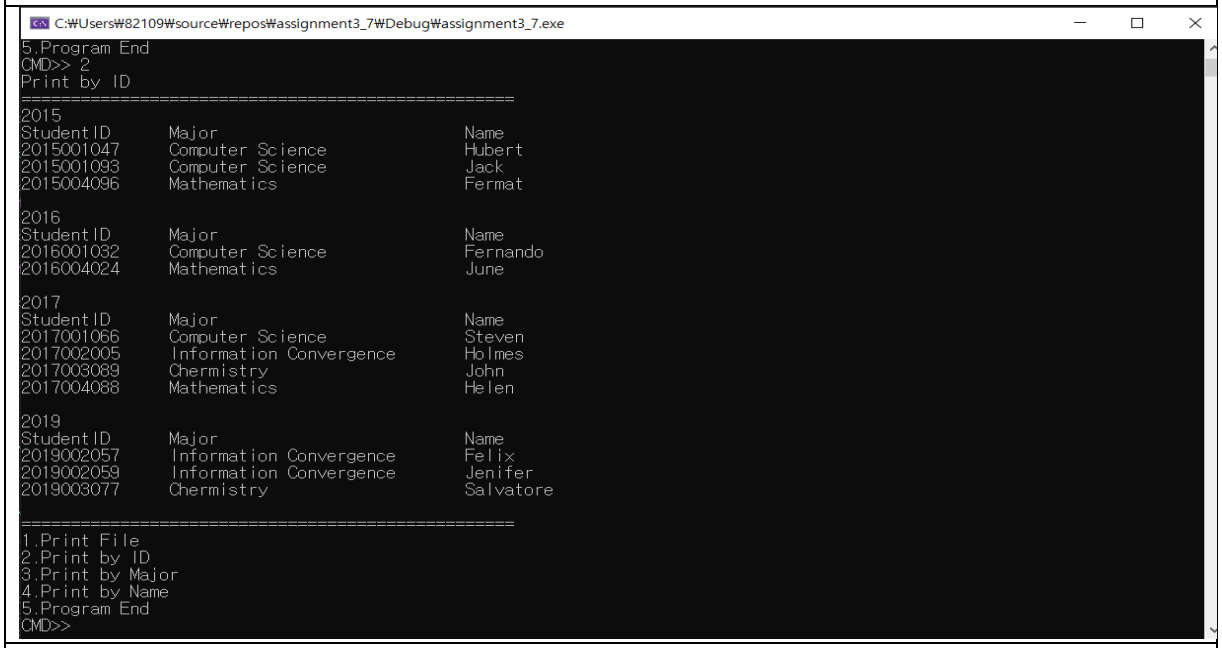
초기화면



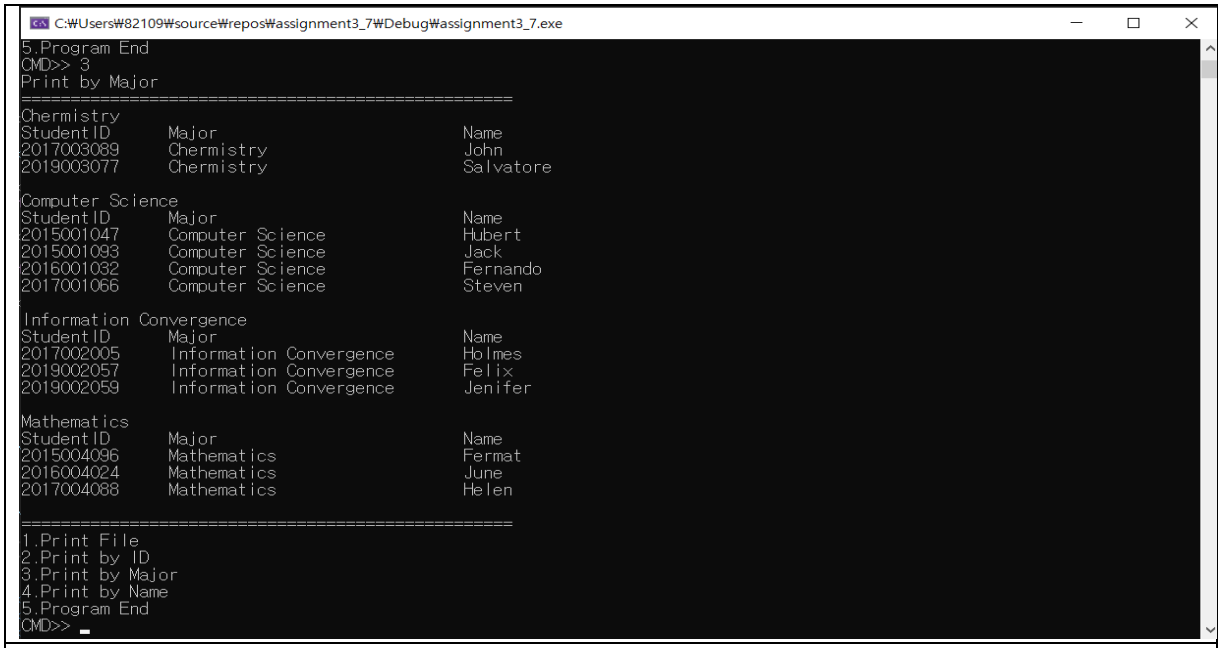
1


C:₩Users₩82109₩source₩repos₩assignment3_7₩Debug₩assignment3_7.exe
1.Print File
2.Print by ID
3.Print by Major
4.Print by Name
5.Program End
CMD>> 1
Print File
==========
2015001093 Computer Science Jack
2017003089 Chermistry John
2017001066 Computer Science Steven
2017002005 Information Convergence Holmes
2015004096 Mathematics Fermat
2019002057 Information Convergence Felix
2017004088 Mathematics Helen
2015001047 Computer Science Hubert
2016001032 Computer Science Fernando
2019003077 Chermistry Salvatore
2019002059 Information Convergence Jenifer
2016004024 Mathematics June
==========
1.Print File
2.Print by ID
3.Print by Major
4.Print by Name
5.Program End
CMD>>

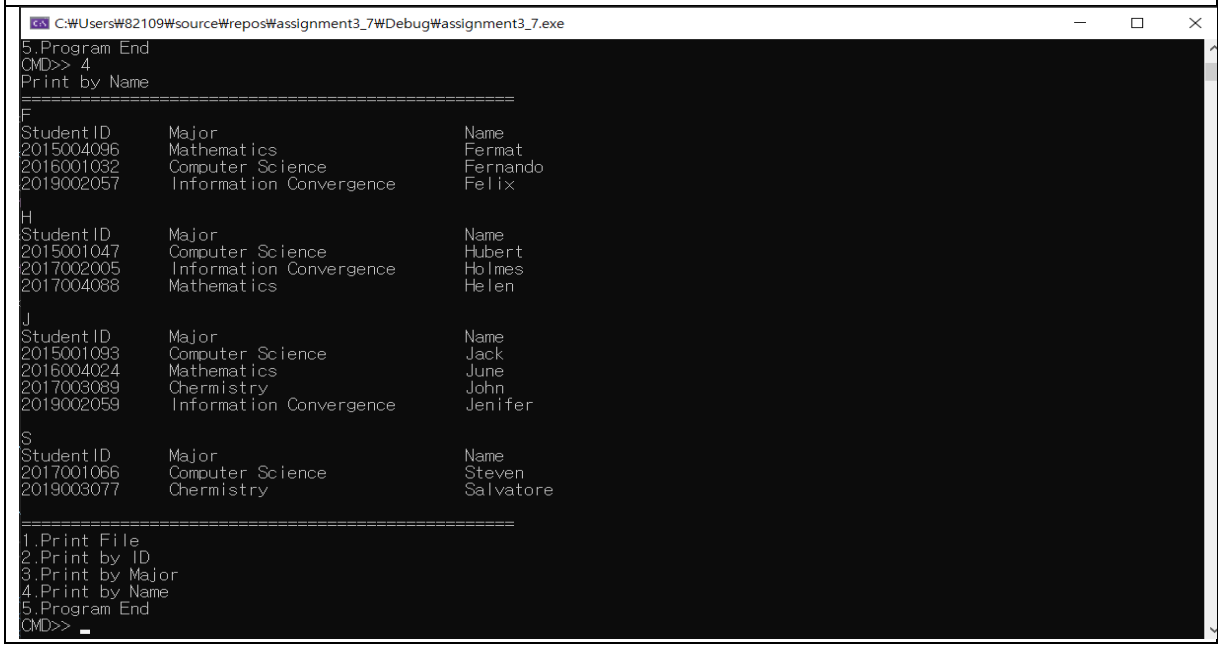
2


C:₩Users₩82109₩source₩repos₩assignment3_7₩Debug₩assignment3_7.exe
5.Program End
CMD>> 2
Print by ID
==========
2015
StudentID Major Name
2015001047 Computer Science Hubert
2015001093 Computer Science Jack
2015004096 Mathematics Fermat
2016
StudentID Major Name
2016001032 Computer Science Fernando
2016004024 Mathematics June
2017
StudentID Major Name
2017001066 Computer Science Steven
2017002005 Information Convergence Holmes
2017003089 Chermistry John
2017004088 Mathematics Helen
2019
StudentID Major Name
2019002057 Information Convergence Felix
2019002059 Information Convergence Jenifer
2019003077 Chermistry Salvatore
==========
1.Print File
2.Print by ID
3.Print by Major
4.Print by Name
5.Program End
CMD>>

3



4



- 고찰

  해당 문제는 2 가지 측면을 공부할 수 있는 과제였다. 첫 번째로 2 차원 Linked List 를 활용하면서 어떻게 Linked List 를 Matrix 처럼 사용할 수 있을 지 알게되었고, 이를 활용할 수 있게되었다. 다음으로 template 의 활용법을 조금이나마 익힌 것이다. template 을 활용하면 중복으로 들어가는 내용을 획기적으로 줄일 수 있다. 다음에 다시 이 과제를 보완할 수 있다면 class 를 template 으로 선언하여 바꾸어보고 싶다.