|  |  |  |
| --- | --- | --- |
| **C++ Programming** | **Student**  **number** | **21600193** |
| **Homework 4** | **Name** | Kim, Hyo Rim |

**1. Comment**

1) struct book{}: Make the information of the book to be saved into struct format. ‘title’, ‘p\_Year’, ‘author’, ‘edition’, ‘b\_Person’ are declared as string type. Declare ‘l\_Day’ and ‘index’ as int. At this time, ‘b\_Person’ and ‘l\_Day’ specify default values. int index represents the number of stored books.

2) int main():Declare b\_List, the pointer type of struct book. Use the 'load\_m' function to store the contents of the file in b\_List. Use the while statement to continue the menu until the 'exit' command is called. Each command uses a switch to call the required function. In this case, the command uses the touuper function so that it is not case sensitive.

3) void printMenu(): Show the menu.

4) string \* p\_Arg(string line): Divide the input command so that it can be used in the program and save it.

5) book \*load\_m(string FileName): It receives the file name and stores the file information in a struct book. At this point, struct book uses static variables to make the saved book available to other functions as well.

6) void save\_m(string fileName, book \* bookList): Save the struct book as a file name.

7) void print\_m(book \* bookList): Receives a struct book and prints out the Book Catalog and a statement specifying the return time limit for the book.

8) book \* insert\_m(book \* bookList, string \* arg): The new book information inputted from the user through the command is stored in the existing information and outputted.

9) book \*lend\_m(book \* bookList, string \* arg): It stores information of a book borrowed by another person.

10) book \*passDay\_m(book \* bookList): Subtract 1 from 'l\_Day' of the book where 'b\_Person' is not “None”.

11) book \* returned\_m(book \* bookList, string \* arg): When the returned book is entered, the 'l\_Day' and 'b\_Person' of the book are returned to their default values.

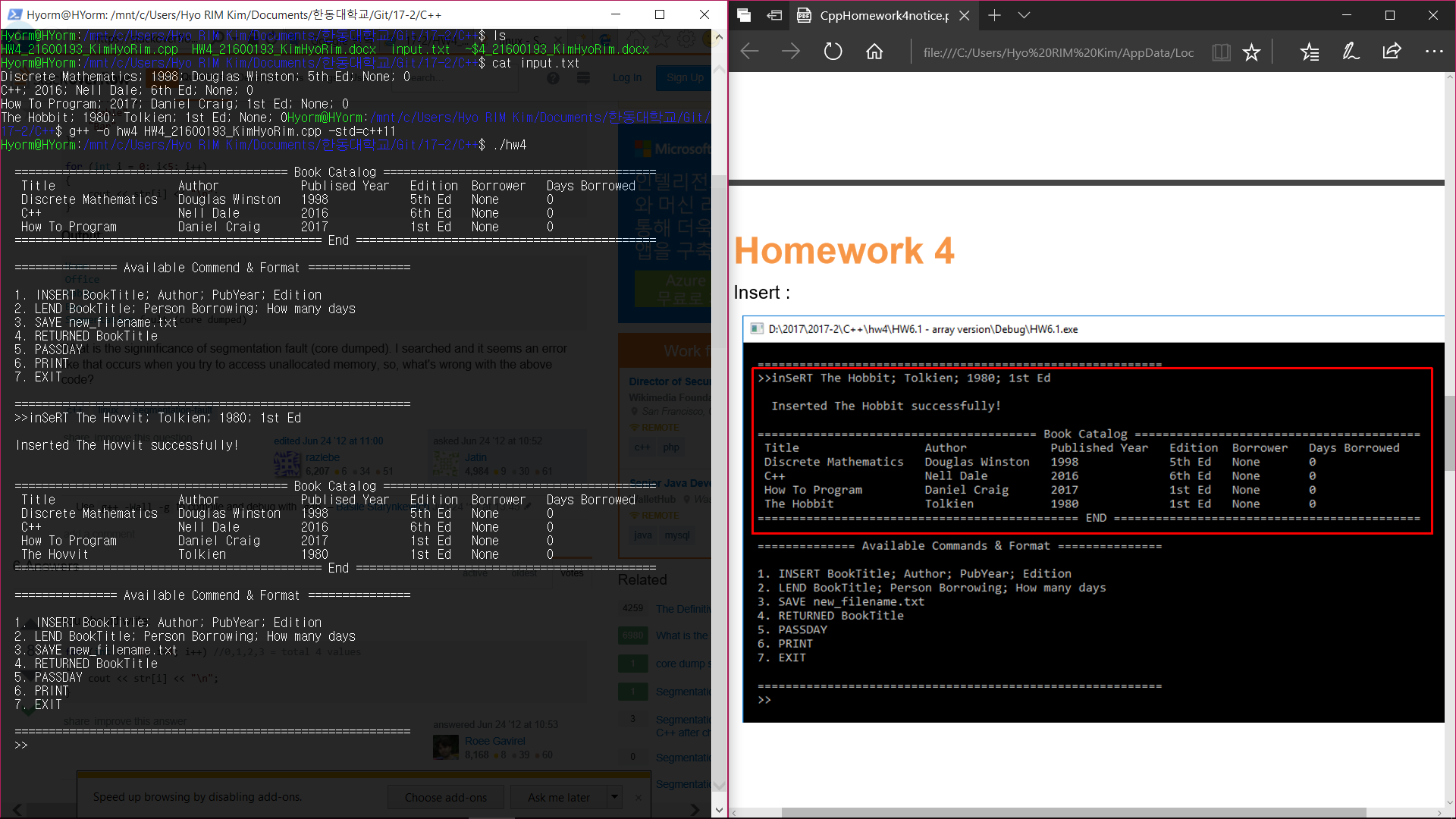
**2. Code**

|  |
| --- |
| #include <iostream>  #include <string>  #include<iomanip>  #include <fstream>  using namespace std;  //DONE: ONE - make  struct book{  string title;  string p\_Year;  string author;  string edition;  string b\_Person = "None";  int l\_Day = 0;  int index;  };  void printMenu();  string \* p\_Arg(string line);  book \*load\_m(string FileName);  void save\_m(string fileName, book \* bookList);  void print\_m(book \* bookList);  book \* insert\_m(book \* bookList, string \* arg);  book \*lend\_m(book \* bookList, string \* arg);  book \*passDay\_m(book \* bookList);  book \* returned\_m(book \* bookList, string \* arg);  int getExit();  int main(){  int setExit = 1;  book \* b\_List;  string openFile = "input.txt";  b\_List = load\_m(openFile);  while(setExit){  string argLine, \* arg;  printMenu();  getline(cin, argLine);  arg = p\_Arg(argLine);  char n1 = arg[0][0];  char n2 = arg[0][1];  switch(toupper(n1)){  case 'I':  b\_List = insert\_m(b\_List, arg);  break;  case 'L' :  b\_List = lend\_m(b\_List, arg);  break;  case 'S':  save\_m(arg[1], b\_List);  break;  case 'R':  b\_List = returned\_m(b\_List, arg);  break;  case 'P':  switch (toupper(n2)) {    case 'A':  b\_List = passDay\_m(b\_List);  break;  case 'R':  print\_m(b\_List);  break;  }  break;  case 'E':  setExit = getExit();  break;  }  }  return 0;  }  void printMenu(){    cout << " =============== Available Commend & Format ===============" << endl << endl;  cout << " 1. INSERT BookTitle; Author; PubYear; Edition" << endl;  cout << " 2. LEND BookTitle; Person Borrowing; How many days" << endl;  cout << " 3. SAVE new\_filename.txt" << endl;  cout << " 4. RETURNED BookTitle" << endl;  cout << " 5. PASSDAY" << endl;  cout << " 6. PRINT" << endl;  cout << " 7. EXIT" << endl << endl;  cout << " ==========================================================" << endl;  cout << " >>";  }  string \* p\_Arg(string line){    int f\_col, i = 0;    static string arg[6];  while(1){  if (i == 0) {  f\_col = line.find(" ");  arg[i] = line.substr(0, f\_col);  line.erase(0, f\_col + 1);  i++;  }  else {  if (line.length() > 0) {  f\_col = line.find(";");  if (f\_col == -1) {  arg[i] = line;  break;  }  arg[i] = line.substr(0, f\_col);  line.erase(0, f\_col + 2);  i++;  }  else  break;  }  }  return arg;  }  //DONE: THREE - function Load  book \*load\_m(string FileName){  //open file and store list  static book bookList[50];  int i = 0;  string line;  ifstream inData;  inData.open(FileName.c\_str());  if(!inData){    cout << FileName << " does not exist." << endl;    }  else{  while(!inData.eof()){    getline(inData, line);  int f\_col, j = 0;  string arr[6];    while(1){  if (line.length() > 0) {  f\_col = line.find(";");  if (f\_col == -1) {  arr[j] = line;  break;  }  arr[j] = line.substr(0, f\_col);  line.erase(0, f\_col+2);  j++;  }  else  break;    }  bookList[i].title = arr[0];  bookList[i].p\_Year = arr[1];  bookList[i].author = arr[2];  bookList[i].edition = arr[3];  bookList[i].b\_Person = arr[4];  bookList[i].l\_Day = stoi(arr[5]);    i++;  }  i--;  bookList[0].index = i;  print\_m(bookList);  }  inData.close();  return bookList;  }  //DONE: THREE - function Save  void save\_m(string fileName, book \* bookList){  // use ofstream and save text file  ofstream outData;  int i = 0;  outData.open(fileName.c\_str());  while (i < bookList[0].index - 1) {  outData << bookList[i].title << "; " << bookList[i].p\_Year << "; " << bookList[i].author << "; "<< bookList[i].edition << "; "<< bookList[i].b\_Person << "; "<< bookList[i].l\_Day << endl;    i++;  }  outData << bookList[i].title << "; "<< bookList[i].p\_Year << "; "<< bookList[i].author << "; "<< bookList[i].edition << "; " << bookList[i].b\_Person << "; "<< bookList[i].l\_Day;  outData.close();  }  //DONE: THREE - function Print  void print\_m(book \* bookList){  string pass[50];  string today[50];  int pass\_j = 0, today\_j = 0;  cout << endl << " ======================================== Book Catalog ========================================" << endl;  cout << setw(8) << "Title" << setw(24) << "Author" << setw(25) << "Publised Year" << setw(10) << "Edition" << setw(10) << "Borrower" << setw(16) << "Days Borrowed" << endl;  for(int i = 0; i < bookList[0].index ; i++ ){  if(bookList[i].b\_Person != "None"){  if(bookList[i].l\_Day == 0){  today[today\_j] = bookList[i].title;  today[today\_j+1] = bookList[i].b\_Person;  today\_j += 2;  }  else if(bookList[i].l\_Day <0){  pass[pass\_j] = bookList[i].title;  pass[pass\_j+1] = bookList[i].b\_Person;  pass\_j += 2;  }  }  cout << " " << bookList[i].title << setw(23 - bookList[i].title.length() + bookList[i].author.length()) << bookList[i].author << setw(22 - bookList[i].author.length()) << bookList[i].p\_Year << setw(22 - bookList[i].p\_Year.length()) << bookList[i].edition << setw(9 - bookList[i].edition.length() + bookList[i].b\_Person.length()) << bookList[i].b\_Person << setw(12 - bookList[i].b\_Person.length()) << bookList[i].l\_Day << endl;  }  cout << " ============================================= End ============================================" << endl;  for(int i = 0; i < pass\_j ; i += 2)  cout << pass[i]<< " SHOULD HAVE BEEN RETURNED ALREADY by " << pass[i+1]<< endl;    for(int i = 0; i < today\_j ; i += 2)  cout << today[i]<< " should be returned today by " << today[i+1]<< endl;  cout << endl;  }  //DONE: THREE - function Insert  book \* insert\_m(book \* bookList, string \* arg){  bookList[bookList[0].index].title = arg[1];  bookList[bookList[0].index].author = arg[2];  bookList[bookList[0].index].p\_Year = arg[3];  bookList[bookList[0].index].edition = arg[4];  bookList[0].index++;  cout << endl << " " << "Inserted " << arg[1] << " successfully!" << endl << endl;  print\_m(bookList);    return bookList;    }  //DONE: THREE - function Lend  book \*lend\_m(book \* bookList, string \* arg){  int j = 0;  for(int i = 0; i < bookList[0].index; i++){  if(bookList[i].title == arg[1])  j++;  }  if(j == 0){  cout << "NO SUCH BOOK!" << endl;  return bookList;  }  cout << endl <<" ==================================== Books Cuurrently Lent ====================================" <<endl;  cout << setw(8) << "Title" << setw(24) << "Author" << setw(25) << "Publised Year" << setw(10) << "Edition" << setw(10) << "Borrower" << setw(16) << "Days Borrowed" << endl;    for(int i = 0; i < bookList[0].index; i++){  if(bookList[i].title == arg[1]){  bookList[i].b\_Person = arg[2];  bookList[i].l\_Day = stoi(arg[3]);  }  if(bookList[i].b\_Person != "None")  cout << " " << bookList[i].title << setw(23 - bookList[i].title.length() + bookList[i].author.length()) << bookList[i].author << setw(22 - bookList[i].author.length()) << bookList[i].p\_Year << setw(22 - bookList[i].p\_Year.length()) << bookList[i].edition << setw(9 - bookList[i].edition.length() + bookList[i].b\_Person.length()) << bookList[i].b\_Person << setw(12 - bookList[i].b\_Person.length()) << bookList[i].l\_Day << endl;    }  cout << " ============================================= End =============================================" << endl << endl;  return bookList;  }  //DONE: THREE - function Passday  book \*passDay\_m(book \* bookList){  cout << endl <<" ==================================== Books Cuurrently Lent ====================================" <<endl;  cout << setw(8) << "Title" << setw(24) << "Author" << setw(25) << "Publised Year" << setw(10) << "Edition" << setw(10) << "Borrower" << setw(16) << "Days Borrowed" << endl;  string pass[50];  string today[50];  int pass\_j = 0, today\_j = 0;  for(int i = 0; i < bookList[0].index; i++){  if(bookList[i].b\_Person != "None"){    bookList[i].l\_Day--;  if(bookList[i].l\_Day == 0){  today[today\_j] = bookList[i].title;  today[today\_j+1] = bookList[i].b\_Person;  today\_j += 2;  }  else if(bookList[i].l\_Day <0){  pass[pass\_j] = bookList[i].title;  pass[pass\_j+1] = bookList[i].b\_Person;  pass\_j += 2;  }  cout << " " << bookList[i].title << setw(23 - bookList[i].title.length() + bookList[i].author.length()) << bookList[i].author << setw(22 - bookList[i].author.length()) << bookList[i].p\_Year << setw(22 - bookList[i].p\_Year.length()) << bookList[i].edition << setw(9 - bookList[i].edition.length() + bookList[i].b\_Person.length()) << bookList[i].b\_Person << setw(12 - bookList[i].b\_Person.length()) << bookList[i].l\_Day << endl;    }  }  cout << " ============================================= End =============================================" << endl;  for(int i = 0; i < pass\_j ; i += 2)  cout << pass[i]<< " SHOULD HAVE BEEN RETURNED ALREADY by " << pass[i+1]<< endl;  for(int i = 0; i < today\_j ; i += 2)  cout << today[i]<< " should be returned today by " << today[i+1]<< endl;  cout << endl;  return bookList;  }  //DONE: THREE - function Returned  book \* returned\_m(book \* bookList, string \* arg){    int j = 0;  for(int i = 0; i < bookList[0].index; i++){  if(bookList[i].title == arg[1]){  if(bookList[i].b\_Person == "None"){  cout << endl<< "ATTENTION No one borrowed that book!!" << endl;  return bookList;  }  else{  j++;  cout << endl << " ========================================= Book Retured ========================================" <<endl;  cout << setw(8) << "Title" << setw(24) << "Author" << setw(25) << "Publised Year" << setw(10) << "Edition" << setw(10) << "Borrower" << setw(16) << "Days Borrowed" << endl;  bookList[i].b\_Person = "None";  bookList[i].l\_Day = 0;  cout << " " << bookList[i].title << setw(23 - bookList[i].title.length() + bookList[i].author.length()) << bookList[i].author << setw(22 - bookList[i].author.length()) << bookList[i].p\_Year << setw(22 - bookList[i].p\_Year.length()) << bookList[i].edition << setw(9 - bookList[i].edition.length() + bookList[i].b\_Person.length()) << bookList[i].b\_Person << setw(12 - bookList[i].b\_Person.length()) << bookList[i].l\_Day << endl;  cout << " ============================================= End =============================================" << endl << endl;  }  }  }  if(j == 0)  cout << endl<< "ATTENTION No Such Book!"<< endl;  return bookList;  }  //DONE: THREE - fundtion Exit  int getExit(){  return 0;    } |

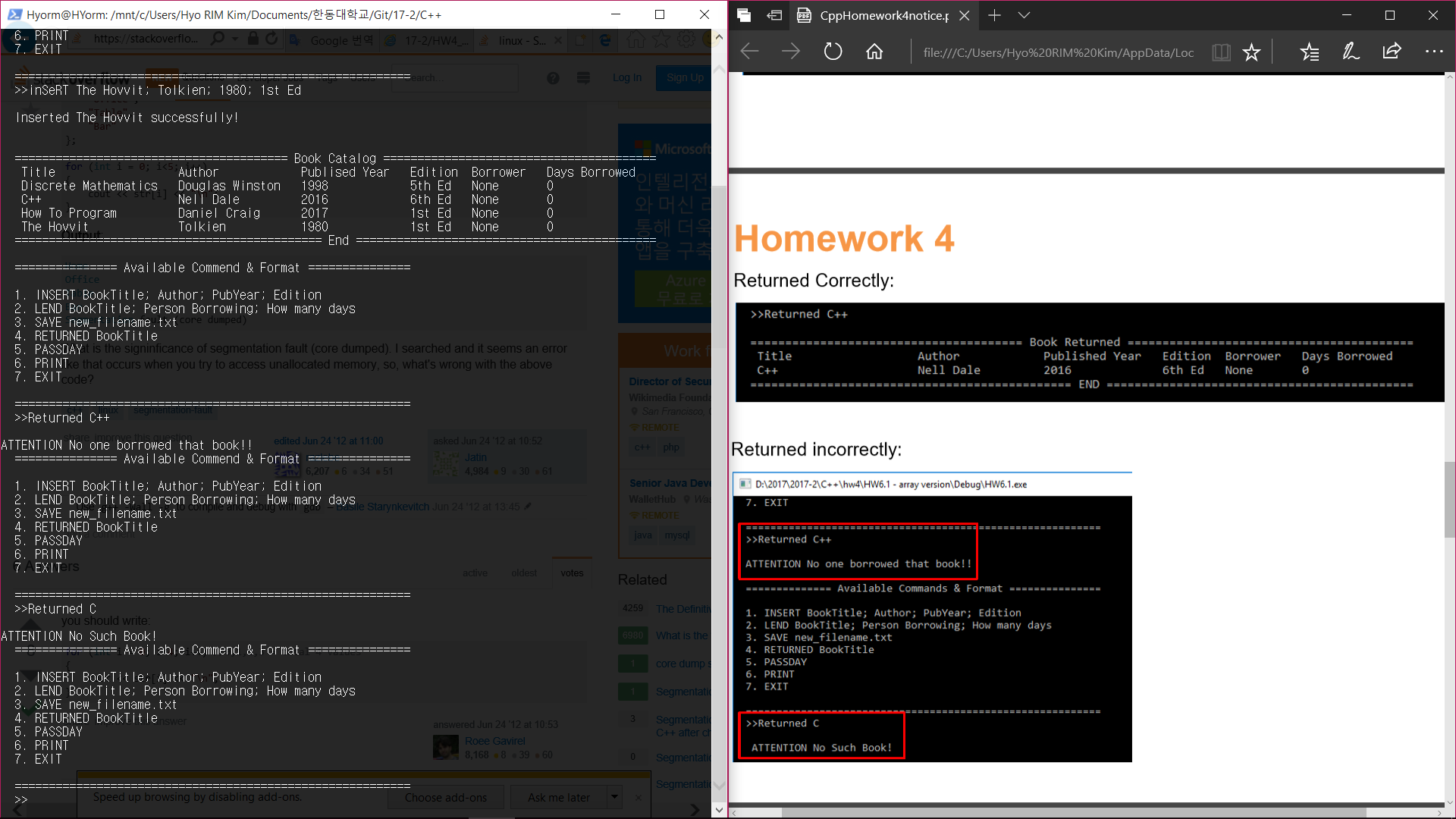
**3. Result**

1)input.txt

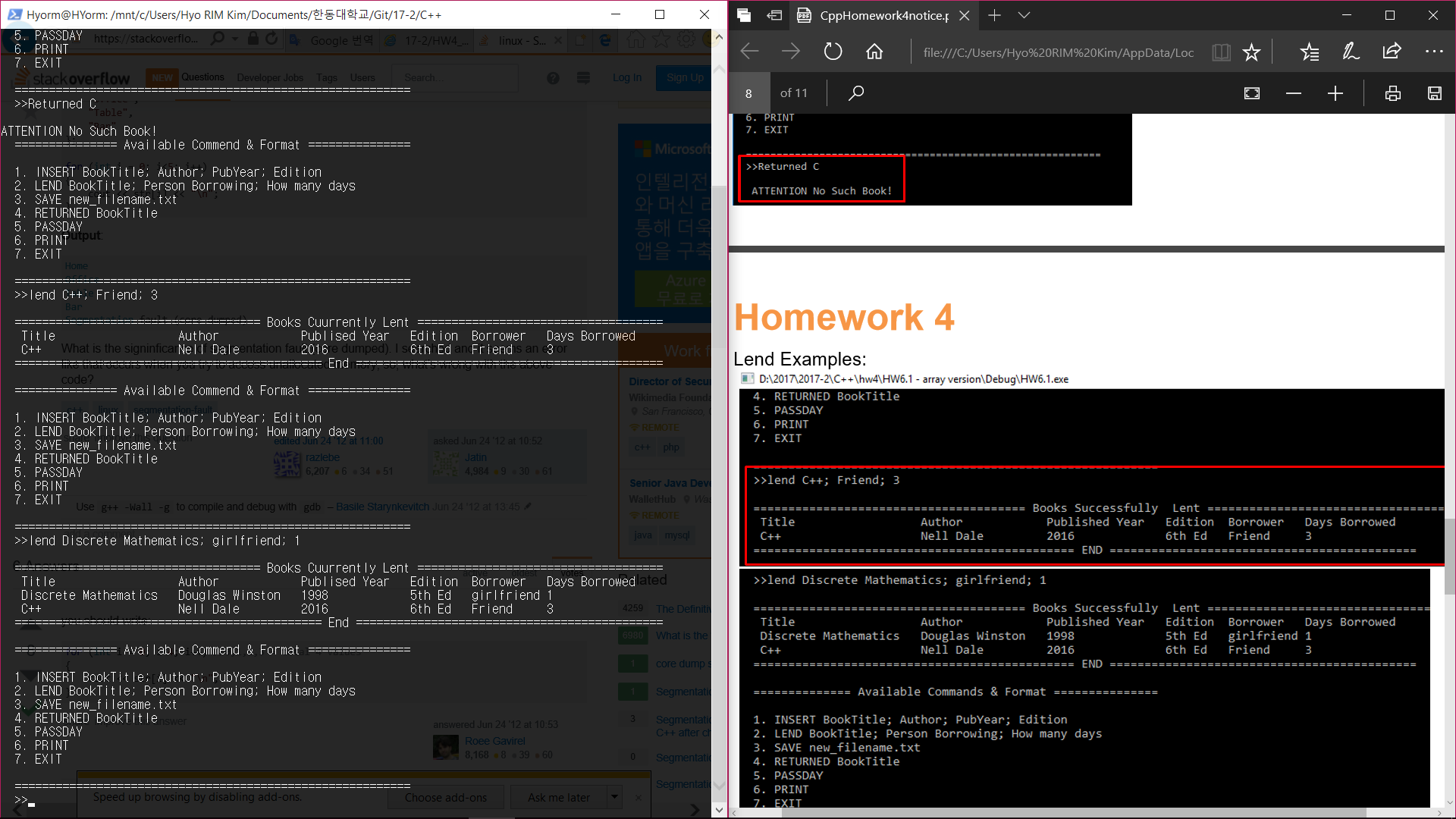
2)Start & Insert



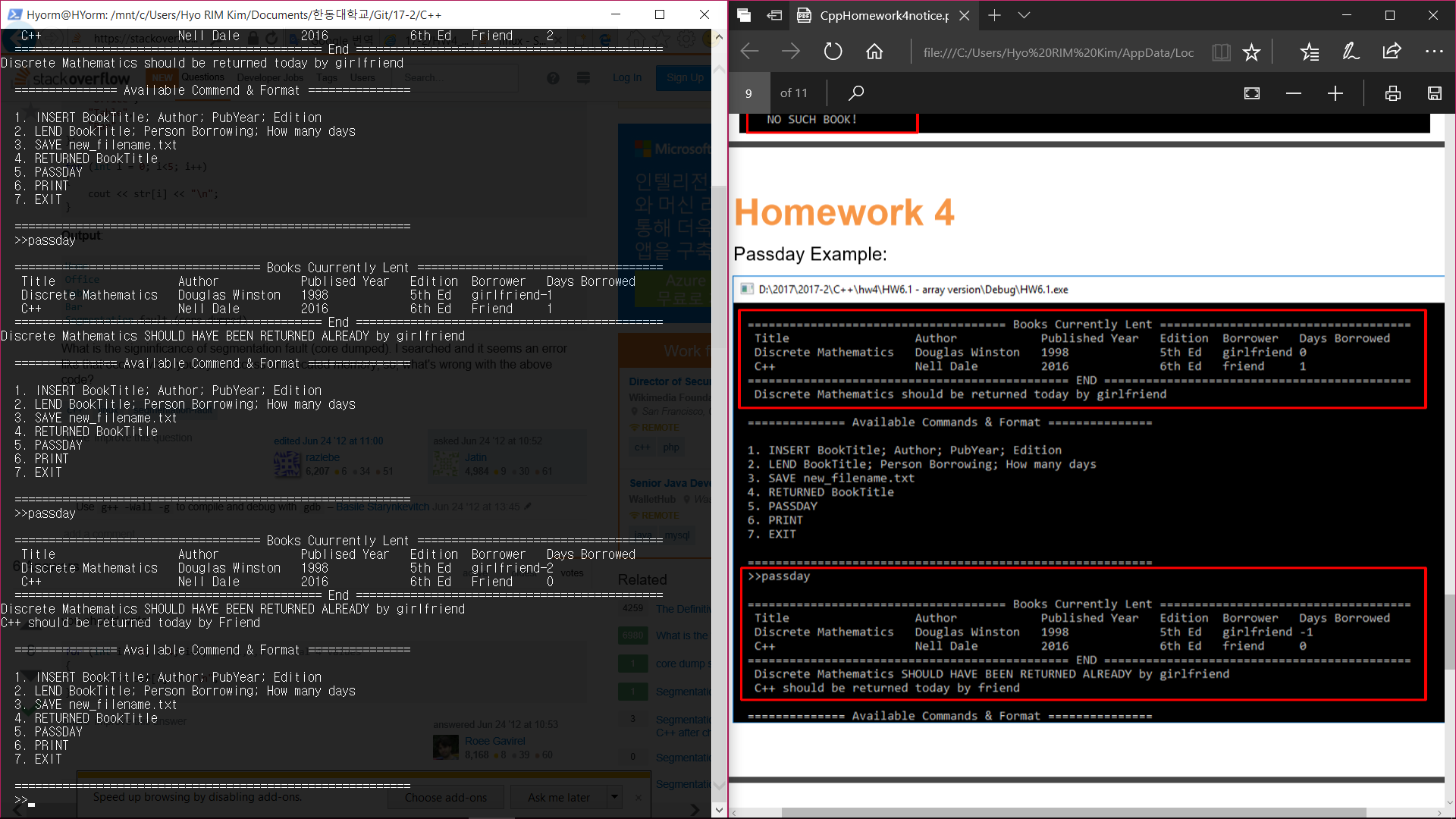
3)Returned



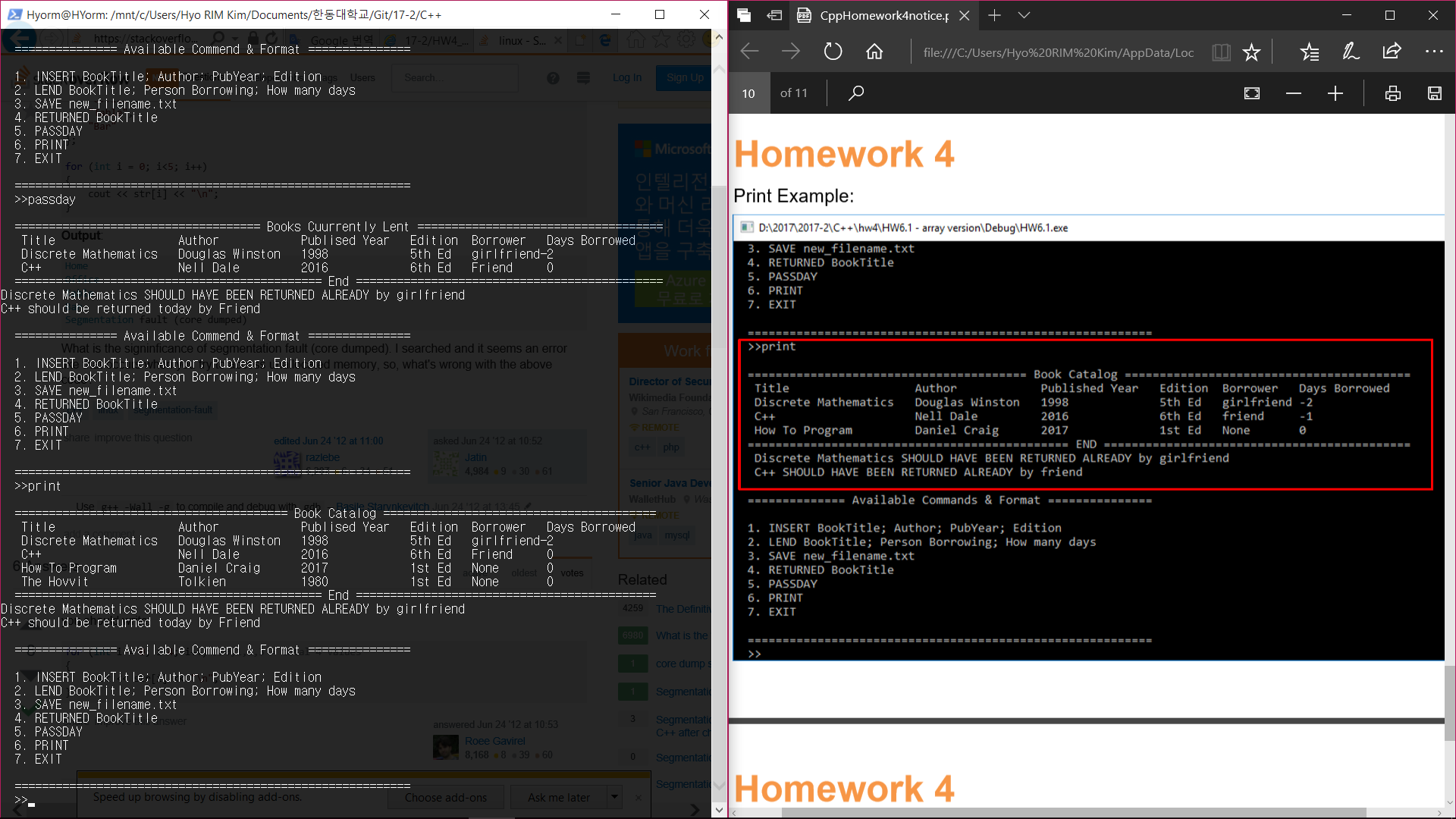
4)Lend



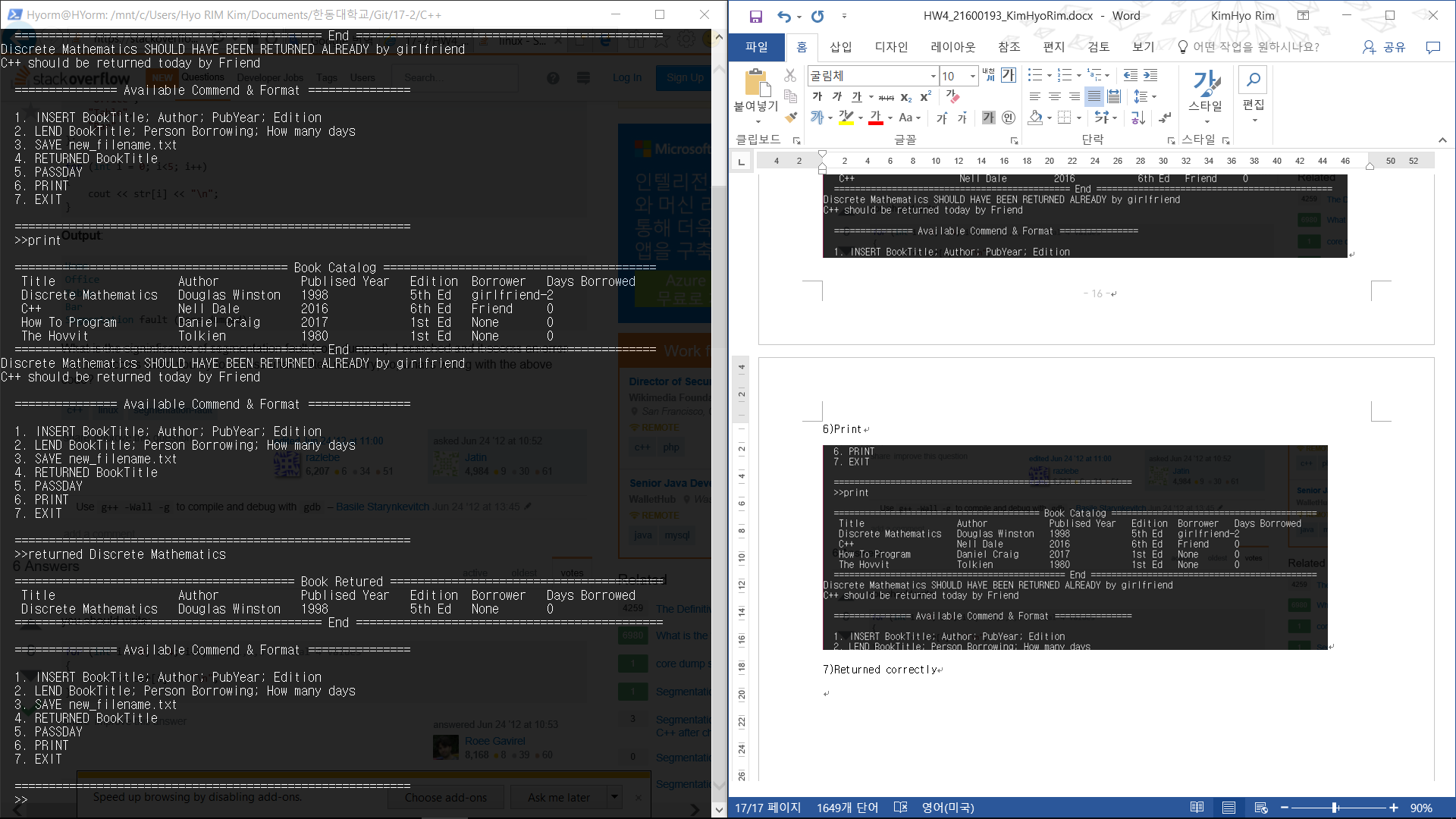
5) Passday



6)Print



7)Returned correctly



8) Save

