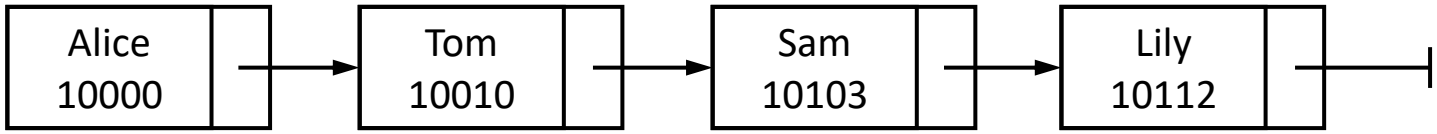
Practice 2. Linked Lists

[CSE2010] Data Structures

Department of Data Science

Overview

- Implement a program for course registration.
- Maintain a linked list of students (with their student IDs and names) who registered a course.
- **The list should be always sorted in the ascending order of student ID.**
- Functions
  - Add a student, given a student ID and name.
  - Delete the student with a given student ID
  - Find a given student ID



Input and Output

- Input file contains a list of following operations.
- Each line represents a single operation.
  - **A<space>[ID]<space>[name]**
    - If [ID] already exists in a list, write a failure message to the output file.
    - Otherwise, write the list after the insertion to the output file.
  - **D<space>[ID]**
    - If [ID] does not exist in the list, write a failure message to the output file.
    - Otherwise, delete the student and write the list after the deletion to the output file.
  - **F<space>[ID]**
    - If [ID] does not exist in the list, write a failure message to the output file.
    - Otherwise, write the student ID and name of the student with that [ID] to the output file.

Input and Output

- Each line represents a single operation.
- Input file

```
A 10003 Mike
A 10000 Alice
A 10010 Tom
A 10010 Tim
F 10001
F 10000
A 10103 Sam
D 10003
D 10003
A 10112 Lily
```

- Output file

```
10003 Mike
10000 Alice 10003 Mike
10000 Alice 10003 Mike 10010 Tom
Addition failed
Search failed
10000 Alice
10000 Alice 10003 Mike 10010 Tom 10103 Sam
10000 Alice 10010 Tom 10103 Sam
Deletion failed
10000 Alice 10010 Tom 10103 Sam 10112 Lily
```

Running Example

- Please refer to the last practice on file I/O to implement this exercise.
- You can implement your code based on the skeleton code provided.

```
[hjkim@localhost linked_lists]$ make all
g++ -std=c++11 -o course course.cpp
[hjkim@localhost linked_lists]$ make run
./course input.txt output.txt
[hjkim@localhost linked_lists]$ cat input.txt
A 10003 Mike
A 10000 Alice
A 10010 Tom
A 10010 Tim
F 10001
F 10000
A 10103 Sam
D 10003
D 10003
A 10112 Lily
[hjkim@localhost linked_lists]$ cat output.txt
10003 Mike
10000 Alice 10003 Mike
10000 Alice 10003 Mike 10010 Tom
Addition failed
Search failed
10000 Alice
10000 Alice 10003 Mike 10010 Tom 10103 Sam
10000 Alice 10010 Tom 10103 Sam
Deletion failed
10000 Alice 10010 Tom 10103 Sam 10112 Lily
```

File I/O in Python

- Open and close a file
  - f = open(pathToFile, mode)
  - f.close();
- But try to use context manager
  - with open(pathToFile, mode) as f:
- Read all the lines in a file
  - lines = f.readlines()
- Write a file
  - f.write()
- Execute
  - python io.py input.txt output.txt

```
import sys

def execute(inFile, outFile):
  lines = inFile.readlines()
  for line in lines:
    words = line.split()
    cap = float(words[1]) * float(words[2])
    outFile.write(words[0] + " " + str(cap) + "\n")

if __name__ == "__main__":
  if len(sys.argv) != 3:
    print("[exe] [input_file] [output_file]")
    sys.exit(1)
  with open(sys.argv[1], "r") as i, open(sys.argv[2], "w") as o:
    execute(i, o)
```

File I/O in C++

- Open a file
  - ifstream file; //or ofstream file;
  - file.open(pathToFile);
- Close a file
  - file.close();
- Read a file
  - getline(file, string);
- Write a file
  - file <<
- Compile and Execute
  - g++ -std=c++11 –o cio io.cpp
  - ./cio input.txt output.txt

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
int main(int argc, char* argv[]){
  if (argc != 3) {
    cerr<<"[exe] [input_file] [output_file]"<<endl;
    exit(1);
  }
  ifstream inFile(argv[1]);
  ofstream outFile(argv[2]);
  string line, ticker;
  double price, cap;
  unsigned long long nShares;
  while(getline(inFile, line)) {
    istringstream iss(line);
    iss >> ticker >> price >> nShares;
    cap = price * nShares;
    outFile << ticker << " " << fixed << cap << endl;
  }
  outFile.close();
  inFile.close();
}
```

File I/O in C

- Open a file
  - FILE* fp = fopen(pathToFile, mode);
- Close a file
  - fclose(fp);
- Read a file
  - fscanf(fp, format, variables);
  - (alternatively, use function getline)
- Write a file
  - fprintf(fp, format, variables)
- Compile and Execute
  - gcc –o cppio io.c
  - ./cppio input.txt output.txt

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[]) {
  if (argc != 3) {
    printf("[exe] [input_file] [output_file]\n");
    exit(1);
  }
  FILE* inPtr = fopen(argv[1], "r");
  if (inPtr == NULL) {
    printf("no such file %s\n", argv[1]);
    exit(1);
  }
  FILE* outPtr = fopen(argv[2], "w");
  if (outPtr == NULL) {
    printf("no such file %s\n", argv[2]);
    exit(1);
  }
  char ticker[256];
  double price, cap;
  unsigned long long nShares;
  while (fscanf(inPtr, "%s %lf %llu", ticker, &price, &nShares) == 3) {
    cap = price * nShares;
    fprintf(outPtr, "%s %f\n", ticker, cap);
  }
  fclose(outPtr);
  fclose(inPtr);
}
```

Submission Guideline

- Submission: source code, sample input & output files
  - Where: Assignments in LMS
  - Deadline: **23:59, March. 20th (Sunday)**
- Extra points
  - **From March 21th (Monday)**
  - Share your code, or input & output on Open Board in LMS.
  - Review classmates' code. Give some questions or comments on his/her post.
  - Answer others' questions on your post.
  - Title: [Practice1]StudentID
    - [Practice1]2021000000