Data Structures and Algorithms

Home Work 05 **Marks 10**

Instructions

Work on this home work individually. Absolutely NO collaboration is allowed. Any traces of plagiarism would result in a ZERO marks in this homework and possible disciplinary action.

Due Date

Paste the solution(s) folder of the problems (source code .cpp files only) labeled with your complete roll number in ITM – HW 05 and ITA – HW 05 folders for IT Morning and IT Afternoon sections respectively till Tuesday, May 21, 2019 before 02:00 PM. These folders are available at \\printsrv\Teacher Data\Umair Babar\Students.

Student Record Database

A **database** is a collection of related pieces of information that is organized for easy retrieval. The following sets of student's account records, for instance, form a **records** database.

Record #	Student ID	First name	Last name	Program	CGPA
0	1001	Ahamd	Raza	BSCS	3.98
1	1002	Shahid	Farid	BSCS	3.96
2	5006	Hassan	Khan	BSCS	3.59
3	1006	Ahmad	Ghazali	BSCS	3.75
4	3007	Asif	Ali	BSSE	3.58
5	1008	Asim	Ali	BSIT	3.65
6	2009	Asim	Rasool	MCS	3.25
7	1000	Ejaz	Ashraf	MCS	3.27
8	8001	Tariq	Butt	BE	3.89
9	1004	Hamid	Ch.	MSc	3.55
10	1003	Syed	Umair	BSIT	3.97
11	7004	Hassan	Ali	BSIT	3.23
12	9001	Imran	Khalil	BSIT	3.19
13	1005	Aman	Ullah	BSIT	3.29
14	8007	Afnan	Ahmad	BSIT	3.45
15	1013	Junaid	Ahmad	BSIT	3.66

Each record in the **records** database is assigned a **record number** based on that record's relative position within the database file. You can use a **record number** to retrieve an account record directly, much as you can use an array **index** to reference an array data item directly. Record numbers are assigned by the database file mechanism and are not part of the account information. As a result, they are not meaningful to database users. These users require a different record retrieval mechanism, one that is based on a **Student ID** (the key for the database) rather than a **record number**.

```
//Database file records
struct AccountRecord
{
	int recNum;			//Record number
	int acctID;			//Student identifier
	char firstName[10];		//First name
	char lastName[10];		//Last name
	char program[10];		//degree program of a student
	double cgpa;			//Student's CGPA
};
```

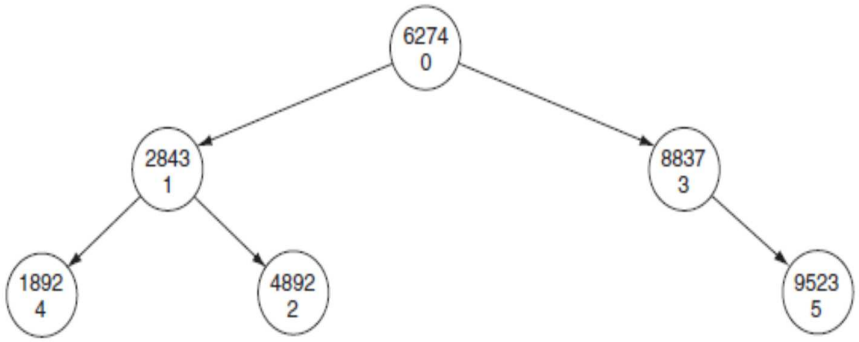
The data of the **account holder(s)** is placed in a **binary file** named **accounts.dat** exactly in the format given in the **AccountRecord** structure.

Retrievals based on **account ID** require an **index** that associates each **account ID** with the corresponding **record number**. You can implement this index using a **binary search tree** in which each data item contains two fields: an **account ID (the key)** and a **record number**.

```
struct IndexEntry
{
	int acctID;		//(Key) Account identifier
	long recNum;		//Record number

	//Return key field
	int getKey () const
	{
		return acctID;
	}
};
```

You build the **index** by reading through the database account by account, inserting successive **(account ID, record number)** pairs into the tree as you progress through the file. The following index tree, for instance, was produced by inserting the account records shown above into an (initially) empty tree.



Given an **account ID**, retrieval of the corresponding account record is a **two-step process**:

1. You retrieve the data item from the **index tree** that has the specified **account ID**.
2. Using the **record number** stored in the **index** data item, you read the corresponding **account record** from the database file.

The result is an efficient retrieval process that is based on **account ID**.

Create a program that builds an **index tree** for the accounts database in the file **records.dat**. Once the index is built, your program should

- Output the account **IDs** in **ascending order**
- Read an account **ID** from the keyboard and output the corresponding account record.

NOTE: - No submission will be accepted after the DUE DATE and TIME.

B E S T O F L U C K