ISRA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
SEMESTER EXANINATION JANUARY 2020

CSDS 422	Data Structures & Algorithms (LAB)	PRACTICAL

INSTRUCTIONS

1. **Attempt questions upto 100 marks.**
2. **It is a take-home open-book exam. You have to submit your solutions within 24 hours on the moodle as a single ZIP file containg solutions to all the questions you attempted. Make sure that the file is open-able and not corrupt.**
3. **You are not stopped to discuss the paper with your fellow students and others, however it is expected that you will abide by the Honor Code, define ethical limits and self-judge yourself to refrain within limits.**
4. **The nature of the problems is such that it requires creativity and original work, so similarity between any two solutions should only be accidental; unless they are copied version of each another.**
5. **A viva-voce conducted after the exam will give you an opportunity to defend your solution and its ownership.**
6. **Paper seems lengthy, but it is do-able in the given timeframe.**
7. **An online discussion session to address any of your queries and concerns is scheduled at 3pm on the BigBlueButton / moodle.**
8. **Download CSDS_Exam.ZIP file from moodle it contains all the projects and code you need for this exam.**

Question # 1. (50 marks)

Download the file Floppy.txt from moodle. It is an image of a 360KB floppy disk formatted with FAT12 filesystem. You may analyze the contents of this file with any HexEdit / HexDump tool. A recent video on the moodle elaborates how to retrieve a file from such a disk image.

Write a program that retrieves the contents of a file from the disk image. In particular, your program should:

a) input a filename eg: "BSICT009.TXT".
b) Open Floppy.txt file and scan through the Root Directory that spans from address 00000A00 to 00001800 and find an entry matching the given filename.
c) If the file is not found, print an error message and exit.
d) Otherwise, retrieve the File Size and the Starting Data Block Number from the Directory entry.
e) Calculate the starting Address for the Data Block and access it by seek-ing its starting position. Print the bytes read from it (be careful, not all bytes in a block may belong to the file, especially the last block of the file).
f) Determine the Next DataBlock from the FAT (located from 00000200 to 000005FF). You may use the following logic for accessing the FAT entry:

```
int AddressOfTheThreeBytes = 0x00000200 + (BlockNumber / 2) * 3;
F.Seek(AddressOfTheThreeBytes, SeekOrigin.FromBegining);
int TwentyFourBits = F.ReadByte() + F.ReadByte()<<8 + F.ReadByte()<<16;
if (BlockNumber%2==0)
{
    NextBlockNumber = TwentyFourBits & 0x000FFF;
}
else
{
    NextBlockNumber = (TwentyFourBits & 0xFFF000)>>12;
}
```

g) repeat steps (e) and (f) for the Next Data Block until you reach the end of the block list which is designated by a 0xFFF value in the NextBlockNumber field.

Question # 2. (20 marks)

LargeUint addition:

Suppose we represent a positive large integer in a Linked List of digits with least significant (ie: units) digit placed at the beginning of the list. Write a function LList<int> Add(LList<int> A, LList<int> B) that Computes and return a third large int as the sum of A and B.

Question # 3. (20 marks)

LargeUint comparision:

Suppose we represent a positive large integer in a Linked List of digits with least significant (ie: units) digit placed at the beginning of the list. We have two such Large Integers A and B and we want to compare them to determine which one is bigger or whether they are equal. Write a function int Compare(LList<int> A, LList<int> B) that returns +1 if A>B, -1 if A<B, and 0 if they are equal.

Question # 4. (30 marks)

In-Order Traversal without recursion:

In the BST class we implemented the In-Order traversal method using recursion technique. Here you are required to implement the same function without using recursion, instead use an explicit Stack. Push node on the stack before moving leftwards. When you have no node to process, pop one from the stack, visit it and move to the right. Stop when the stack is empty.

Question # 5. (40 marks)

HeapSort in file:

Data.BIN file contains 1000 integers stored as 4-bytes little endian. Write a program that implements Heapsort algorithm on file data to sort those numbers. You do not have to load these numbers in memory array.

Question # 6. (30 marks)

LinkedList in arrays:

Consider a datastructure to hold separate transaction lists of some people in an array. An example of such an array is shown on the right, With Ahsan = 3, and Baber = 7 as indexes of their first records, Ahsan's transactions can be traversed as 7, 0, and 5 while Babers transactions are 3, 1, and 4.

	Description	Amount	Next
0	Dinner	300	5
1	Shopping	1400	4
2	Shopping	2000	6
3	Dinner	500	1
4	Lunch	400	-1
5	Tuition Fee	5000	-1
6	Garbage	3243	8
7	Lunch	300	0
8	%sf#dfyu*T	878689	9
9	6dyh2$23t	2242352	-1

Similarly, a List of all the free records is maintained by the variable FreeRec. The first free record in this list is at index 2, the next is at index 6 and so on.

Write Code for the following functions:

```
static void AddRecToFreeList(int RecNo)
static int RemoveARecFromFreeList()
static void AddRecToPersonsList(string PersonName, int RecNo)
static void RemoveRecFromPersonsList(string PersonName, int RecNo)
```

Question # 7. (50 marks)

Event Driven Simulation:

A bank wants to determine how many full-service tellers it should run in order to minimize the average waiting time of its customers. Customer waiting time is the amount of time the customer spends in the queue, waiting for his turn if all tellers are busy.

Given "BankDay.CSV" file containing a list of customer arrival times and their transaction lengths on a typical bank day, you have to compute the average waiting time for the customers if the bank runs N full service tellers, N being input to your program.

For this purpose your program should employ a technique called Event Driven Simulation which works by maintaining a priority queue PQ of time bound events, giving priority to the event having the smallest value of starting-time (ie: the one that should occur prior to all other events in the PQ). Your program will loop on the PQ (until it becomes empty) and remove one event at a time and process it ie: perform necessary actions for that event. The action may result in adding more events in the PQ. You may think of removing an event from the PQ as advancing the clock to that time.

There are mainly two types of events in this problem:

1. Customer Arrival Event
2. Transaction Completion Event

Customer arrival events are all known in advance and can be created and loaded in the priority queue during program initialization. When a customer arrival event (with event time T and customer C) occurs you have to see whether there are any idle tellers, which you can easily find out by maintaining the number of busy tellers B. If there are no free tellers, put the customer C in the Customer Queue Q for waiting. But if there are some idle tellers then advance the customer directly to a teller by performing following 3 steps:

- Note that the Customer's waiting time C.WaitingTime is ZERO.
- Increment the number of Busy Tellers B.
- Create a Transaction Completion Event for time T+C.TransactionTime and put it in the PQ.

When a transaction completion event (with event time T) occurs, see whether there are any customers waiting in the Customer Queue Q. If there are no customers waiting, just decrement the number of busy tellers B. But if there are some customers waiting, remove one customer C from the Customer Queue Q and perform the following 2 steps:

- Note the Customer's waiting time C.WaitingTime as T-C.ArrivalTime.
- Create a Transaction Completion Event for time T+C.TransactionTime and put it in the PQ.

Once you are done with all the events, you can calculate the average waiting time by dividing the sum of waiting times of all the customers by the customer count.

You may have to define the datastructures (ie: classes) for Customer and Event. Event will be an abstract class with two subtypes as indicated above. Define appropriate attributes and methods in these classes.

Question # 8. (20 marks)

SelectionSort in file:

Data.BIN file contains 1000 integers stored as 4-bytes little endian. Write a program that implements SelectionSort algorithm on file data to sort those numbers. You do not have to load these numbers in memory array.

Question # 9. (10 marks)

NumToWords:

Write a function **string NumToWords(int N)** in C# that return the amount in words for any given positive number. Eg: for N=6275183, return "Sixty Two Million Seven Hundred Fifty One Thousand and One Hundred Eighty Three".

Question # 10. (10 marks)

Permutations of a given string:

Write a function **List<string> Permutations(string S)** that returns a list containing all permutations of a given string. E.g: for S="ABC", return ["ABC", "ACB", "BAC", "BCA", "CAB", "CBA"].

---- GOOD LUCK ----