COMP 1603 – Computer Programming III

2021/2022 Semester 2

Coursework Exam

Section 2

Duration: 2 hours (Start at 10:00 a.m. and Finish at 12:00 noon)

This paper has 4 pages and 3 questions.
Each question begins on a separate page and carries 20 marks.
Answer *all* questions.

1. (a) Consider the following declaration of an *Account* struct:

```
struct Account {
	int accNumber;
	double balance;
};
```

The accounts in a certain branch of a bank are stored in the binary file `Accounts.dat`. Bytes 0 to 89 in the file are characters containing branch information such as address, telephone contact, manager's name, etc. The remaining bytes of the file contain the *Account* structs.

Write a *main* function to (i) read all the accounts from the file and (ii) find and display the number of accounts in the branch and the total balance in all the accounts. Only memory address (pointer) variables are permitted except for the file variable. Arrays are **not** permitted.

[10 marks]

(b) Consider the following declaration of a node in a linked list, each of which stores the address of an *Account* struct:

```
struct Node {
	Account * data;
	Node * next;
};
```

(i) Write a function *createNode* which creates a node in a linked list of accounts. It also creates an *Account* struct based on the account data passed as parameters, and then stores the address of the *Account* struct in the newly created node. The function returns the address of the new node. Its heading is:

```
Node * createNode (int accNumber, double balance);
```

[3 marks]

(ii) **Assume** that the function *insertList* is available which accepts as parameters the top of a linked list, an account number, and a balance. The function creates a new node at the top of the linked list and returns the address of the top of the linked list. Its heading is:

```
Node * insertList (Node * top, int accNumber, double balance);
```

Write statements to declare and create a linked list with the following accounts:

accNumber	balance
100	1000.00
200	2000.00

[2 marks]

(iii) The C-string *branchData* has been declared as follows. **Assume** that it contains the appropriate data.

```
char branchData [90];
```

Also, assume that *top* contains the top of a linked list of accounts in a certain branch of the bank. Write statements to save the characters in *branchData* to the binary file `Accounts.dat`, followed by the *Account* data in each node of the linked list of accounts.

[5 marks]

Total Marks for Question 1: 20

Question 2 Starts on Page 3 …

2. (a) Consider the following two sequences of positive integers:

```
1, 5, 9, 13, 17, 21, … , 993, 997, 1001
1, 5, 9, 13, 17, 21, … , 93, 97, 101
```

Write a **recursive** function, *sumSequence*, to find the sum of the sequences. Call *sumSequence* (twice) to find and display the sum of each sequence.

[5 marks]

(b) An integer array *A* contains a person's blood glucose readings taken each day over a period of *n* days. Write a **recursive** function, *biggestChange*, to find the biggest difference in the readings between two adjacent days (ignoring whether the difference is positive or negative). The function has the following heading:

```
int biggestChange (int A[], int n, int biggestSoFar);
```

If *A* contains the values {90, 91, 100, 120, 110, 101, 95}, the call *biggestChange (A, 7, 0)* should return 20, which occurred when the readings went from 100 to 120.

Note 1: The solution can be derived from the *isSorted* recursive function.
Note 2: Use the *fabs* function to convert a negative or positive number to its positive value.
Note 3: The function will be called initially with *biggestSoFar* = 0.

[7 marks]

(c) A node in a linked list of integers is declared as follows:

```
struct Node {
       int data;
       Node * next;
};
```

:

Write a **recursive** function, *deleteRange* which given the address of the top of a linked list, and two integers *m* and *n* as parameters, deletes all the nodes in the linked list which have a value less than *m* or greater than *n*. The function returns the top of the modified linked list. Its heading is:

```
Node * deleteRange (Node * top, int m, int n);
```

NB: Memory must be released when nodes are deleted.

[8 marks]

Total Marks for Question 2: 20

Question 3 Starts on Page 4 …

3. A *SortedSet* is an abstract data type (ADT) for storing a unique set of elements (i.e., no duplicates must be present) in sorted order. A *SortedSet* of integers is declared as follows:

```
#define MAXSET 1000

struct SortedSet {
       int numElements;          // number of elements in the SortedSet
       int elements [MAXSET];    // the actual elements in the SortedSet
}
```

This question requires you to write several ADT operations (functions) for a *SortedSet*. You can **assume** that the following functions have already been written so you can call them if you wish:

Function	**Description**
void insertSS (SortedSet * ss, int key)	Inserts *key* in *ss*. **The function checks for overflow but NOT for duplicates.**
bool containsSS (SortedSet * ss, int key)	Returns *true* if *ss* contains *key* and *false*, otherwise.
void displaySS (SortedSet * ss)	Displays the elements of *ss* on the monitor.
`SortedSet * copySS (SortedSet * ss);`	Creates a copy of the data in *ss* and returns the copy.

(a) Write a function *initSortedSet* which creates, initializes, and returns a new *SortedSet*. Its heading is:

```
SortedSet * initSortedSet ();
```

[1 mark]

(b) Write a function *initSortedSetFromFile* which creates a new *SortedSet*, opens the text file passed as a parameter, inserts the data from the text file in the *SortedSet*, and returns the new *SortedSet*. Its heading is:

```
SortedSet * initSortedSetFromFile (char filename[]);
```

[5 marks]

(c) Write a function *intersectionSS* which creates and returns a *SortedSet* containing the elements that are present in both of the *SortedSet*s passed as parameters. Its heading is:

```
SortedSet * intersectionSS (SortedSet * ss1, SortedSet * ss2);
```

[5 marks]

(d) Write a function *unionSS* which creates and returns a *SortedSet* containing the elements that are present in either one of the *SortedSet*s passed as parameters. Its heading is:

```
SortedSet * unionSS (SortedSet * ss1, SortedSet * ss2);
```

[5 marks]

(e) Write a *main* function to do the following:

- Create a *SortedSet*, *ss1*, from the data stored in the file, `SortedSet1.txt`.
- Create a *SortedSet*, *ss2*, from the data stored in the file, `SortedSet2.txt`.
- Display the elements in *ss1*.
- Display the elements in *ss2*.
- Create a *SortedSet*, *ss3*, containing the elements that are present in both *ss1* and *ss2*.
- Create a *SortedSet*, *ss4*, containing the elements that are present in either *ss1* or *ss2*.
- Display the elements in *ss3*.
- Display the elements in *ss4*.

[4 marks]

Total Marks for Question 3: 20

End of Question Paper