NGEE ANN POLYTECHNIC

FEBRUARY 2020 EXAMINATION

School of InfoComm Technology
(Diploma in Information Security & Forensics)
(Diploma in Information Technology)
(Diploma in Financial Informatics)

Level 2/3

Time Allowed: 2 Hours

DATA STRUCTURES & ALGORITHMS
(011791)

INSTRUCTIONS TO CANDIDATES:

1. **Check carefully to ensure you are sitting for the correct paper.**
2. There are FIVE questions. Answer ALL questions.
3. All questions carry equal marks.
4. Begin each question on a separate page.
5. Appendix 1 shows the `BinaryT` and `BinaryTNode` classes.
6. This paper consists of 9 pages including this cover page. Check carefully to make sure your set is complete.

There are FIVE questions. Answer ALL questions.

QUESTION 1 (20 marks)

(a) Given the following program:

```
void increaseFirst(int *i) {
     (*i)++;
}

void increaseSecond(int i) {
     i++;
}

void increaseThird(int &i) {
     i++;
}

void increaseFourth(int **i) {
     *i = new int(0);
}

void increaseFifth(int *&i) {
     i = new int(69);
}
int main()
{
     int *a = new int(42);
     cout << "Result" << endl;
     increaseFirst(a);
     cout << "first " << *a << endl;
     increaseSecond(*a);
     cout << "second " << *a << endl;
     increaseThird(*a);
     cout << "third " << *a << endl;
     increaseFourth(&a);
     cout << "fourth " << *a << endl;
     increaseFifth(a);
     cout << "fifth " << *a << endl;
}
```

(i) What is the output of the above program?

(5 marks)

(ii) Assuming that the first line in the `main()` function is replaced by the following line:

`int a = 42;`

Make modifications to the code in the `main()` function so that the program would compile and execute to produce the same output as in part (a)(i).

(5 marks)

QUESTION 1 (cont.)

(b) Figure 1(b) shows a ***List ADT*** specification for a linked list.

```
class List
{
  private:
        struct Node
        {
               int item;
               Node *next;
        };
        Node *front; // first node in list

  public:
        List();
        void sortedInsert(Node newNode);
        // other functions not shown
};
```

Figure 1(b): `List.h` File

Write the implementation of the `sortedInsert()` function that inserts the given `newNode` into the correct position in a linked list sorted in ascending order.

For example, if the linked list has the following values:

`0 2 3 4 6 8 9`

using the `sortedInsert()` function to insert 5 would result in the following linked list:

`0 2 3 4 5 6 8 9`

Note: Duplicate values are allowed in the linked list.

(10 marks)

QUESTION 2 (20 marks)

(a) Figure 2(a) shows a ***Stack ADT*** specification that stores integers.

```
class Stack
{
  private:
       int array[] = new int[10];
       int size = 0;

  public:
       Stack();
       // returns true if stack is empty,
       // otherwise returns false
       bool isEmpty();
       // pops item from the top of the stack
       void pop(int &item);
       // pushes item to the top of the stack
       void push(int item);
       // counts the number of positive and
       // negative integers in the stack
       void countPosNeg(int &pos, int &neg);
};
```

Figure 2(a): `Stack.h` File

(i) Write an implementation of the `countPosNeg()` function that takes two reference parameters, `pos` and `neg`. After the call, `pos` holds the count of positive integers on the stack and `neg` holds the count of negative integers on the stack.

(6 marks)

(ii) Given the following declaration:

```
Stack aStack;
```

Write code that uses the `countPosNeg()` function to count the number of positive integers and negative integers in the stack, `aStack`. After the call, print both values.

(4 marks)

QUESTION 2 (cont.)

(b) A Random Access Queue has the following operations:

(A) `pushRight(x)`: to add a new item, `x`, to the tail

(B) `popLeft()`: to remove the item at the head

(C) `elementAt(i)`: to retrieve the item at position `i` without removing it. `i = 0` gives the item at the head, `i = 1` the following element and so on. If there are less than `i` elements, the value -999 is returned.

(i) Suppose this data structure is implemented using a **linked list**.

State the time complexity of each operation. Explain your answer.

(6 marks)

(ii) Suppose this data structure is implemented using an **array**. Figure 2(b)(ii) shows the queue holding 4 integer items, stored within an array of size 8.

index	0	1	2	3	4	5	6	7
				63	59	22	94	
				`head`			`tail`	

Figure 2(b)(ii): Sample data in array implementation

Write an implementation of the `elementAt()` function which has a time complexity of O(1).

The function prototype of `elementAt()` is

```
int elementAt(int i);
```

You may assume that `size`, `head` and `tail` are attributes of the data structure. The attribute `size` stores the number of elements, the attribute `head` stores the index of the first element; and the attribute `tail` stores the index of the last element in the data structure.

(4 marks)

QUESTION 3 (20 marks)

(a) Suppose a sorted array has been rotated at some pivot point. Figure 3(a)(i) shows an array with sorted elements. After rotating 2 elements, the resulting array is shown in Figure 3(a)(ii).

10	20	30	40	50

Figure 3(a)(i): Array of Sorted Integers

30	40	50	10	20

Figure 3(a)(ii): Array of Rotated Integers

The function `getPivotPt()` returns the pivot point of the array in $O(\log_2 n)$.

For example,

```
getPivotPt(anArray, 5)
```

when applied to the array in Figure 3(a)(ii) returns 3, the pivot point.

The function prototype of `getPivotPt()` is:

```
int getPivotPt(int anArray[], int n);
```

Write a search function `pivotedSearch()` to search for a target in the rotated array with time complexity of $O(\log_2 n)$. It retuns the index of the target if found or -1 if not found. This function must use the `getPivotPt()` function.

The function prototype of `pivotedSearch()` is:

```
int pivotedSearch(int anArray[], int n, int target);
```

Note: You may make use of the `binarySearch()` function:

```
int binarySearch(int anArray[], int start, int end,
                                       int target);
```

(10 marks)

QUESTION 3 (cont.)

(b) Figure 3(b) shows a binary tree.

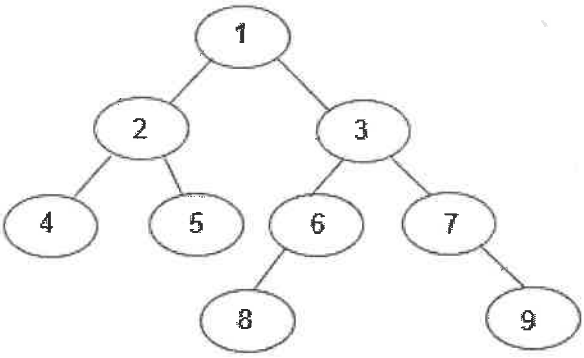

Figure 3(b): A Binary Tree

For the binary tree in Figure 3(b), the ancestors of node 8 are 6, 3 and 1; the ancestors of node 7 are 3 and 1; the ancestor of node 2 is 1. Node 1 has no ancestors.

Implement a ***recursive*** function `printAncestors()` which prints all ancestors of a given node value in it. Nothing is printed if the given node has no ancestors or if the given node is not found in the binary tree. The function returns `true` if the given node is found or `false` if the given node is not found.

The function prototype of `printAncestors()` is

```
bool printAncestors(BinaryTNode* aTree, BinaryTNode node);
```

The `BinaryT` and `BinaryTNode` classes are shown in Appendix 1.

(10 marks)

Note: You need not write any code for Questions 4 and 5.

QUESTION 4 (20 marks)

Consider a set of values 0, 1, 2, 3, …, n and the following data structures:

(A) Binary Search Tree

(B) AVL Tree

(C) Hash Table of size 6 that resolves collisions by chaining

(a) For each of the data structures, draw a diagram of the data structure after the insertion of the set of values up to n = 9.
(10 marks)

(b) Explain clearly how each data structure will evolve for large n values. Derive the worst-case time complexity for each operation of searching for a given value.
(6 marks)

(c) Suppose there are 1000 values in each data structure and a search is required for a target value. Which of the data structures is the most time-efficient? Justify your answer with appropriate calculations.
(4 marks)

QUESTION 5 (20 marks)

Figure 5 shows an array of 9 integers:

5	0	2	4	3	8	7	6	9

Figure 5: Array of Integers

(a) (i) The array is to be sorted using the quicksort algorithm. Which of these array elements: 3, 5, 9 should be the pivot value? Justify your answer.
(4 marks)

(ii) Trace the quicksort algorithm using the pivot from part (a)(i).
(6 marks)

(b) Trace the merge sort algorithm using the array in Figure 5.
(8 marks)

(c) Draw the array in Figure 5 after ONE iteration of the selection sort algorithm.
(2 marks)

Appendix 1: BinaryT and BinaryTNode classes

```
class BinaryT
{
     private:
          BinaryTNode *root;

     public:
          BinaryT(BinaryTNode *n) {  root = n;  }

}; // end of BinaryT class

class BinaryTNode
{
     private:
          int data;
          BinaryTNode *left, *right;

     public:
          BinaryTNode(int d, BinaryTNode *lft,
                                    BinaryTNode  *rght)
          {
               data = d;
               left = lft;
               right = rght;
          }

          int getData() {  return data;  }

          void setData(int d) {  data = d;  }

          BinaryTNode getLeft() {  return *left;  }

          void setLeft(BinaryTNode *n) {  left = n;  }

          BinaryTNode getRight() {  return *right;  }

          void setRight(BinaryTNode *n) {  right = n;  }

          bool isLeaf()
          {
               return (left == NULL) && (right == NULL);
          }

}; // end of BinaryTNode class
```

** END OF PAPER **