Midterm 2018 Solution

Question 1:

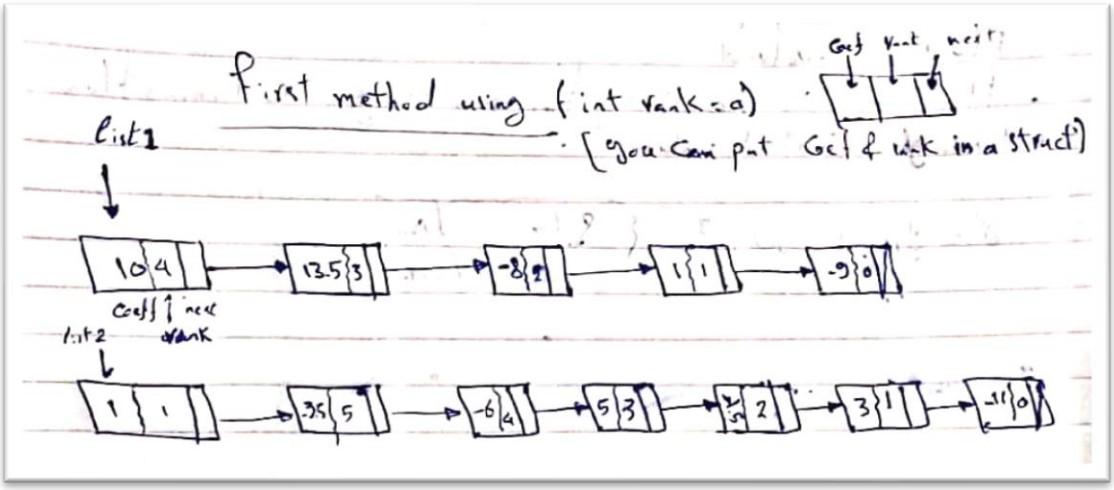
1. Given the two polynomials:
   - $10x4 + 13.5x3 - 8x2 + x - 9$
   - $x6 - 3.5x5 - 6x4 + 5x3 - 1.5x2 + 3x - 11$

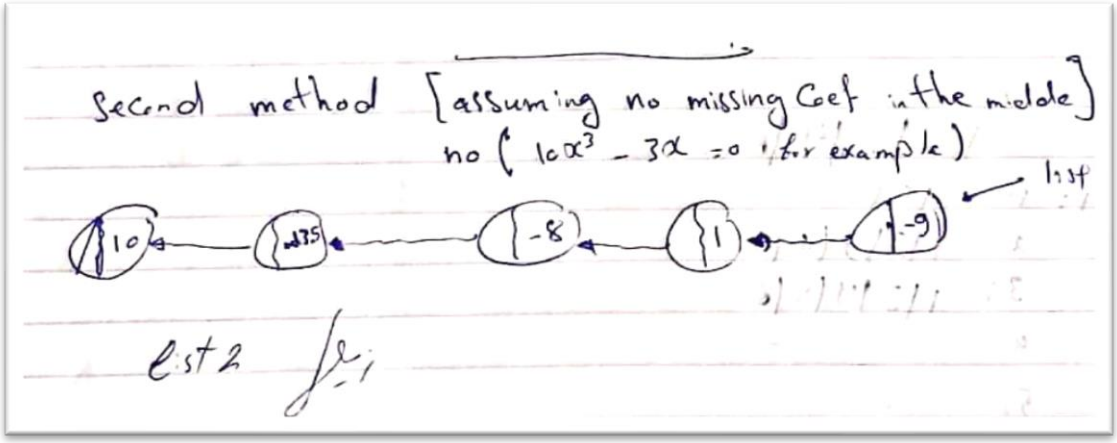
   a) Describe a node structure that enables you to represent a polynomial as a linked list. Use diagrams, pseudo-code, or C++ struct.

   - I'll assume he needs to save the Fn. As it's written, so I'll use struct save coeff.

   ```
   struct node {
           double coeff = 0;
           (optional) int rank=0;
           node *next = nullptr; //pointer to next node
           };
   ```

   b) Using the developed node structure in (a), represent the two given polynomials as two independent linked lists (i.e their layout in memory). Use diagrams.





c) Develop an efficient algorithm to add the given polynomials in linear time O(n). Assume any necessary condition. Use diagrams or pseudo-code.

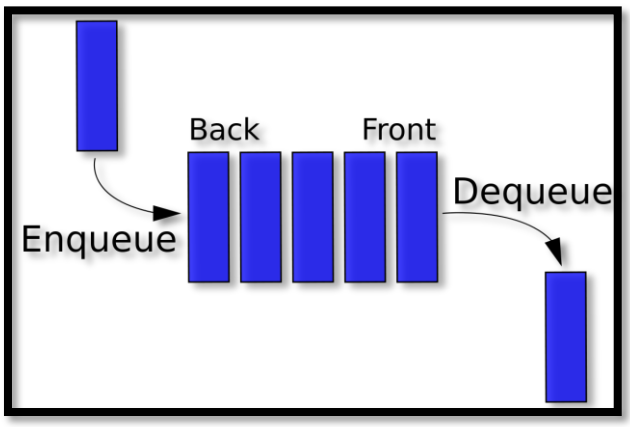
- Using pseudo-code:

```
1   //list1 , list2 are 2 functions
2   double d =0;
3   add(node *list1, node *list2)
4   {
5       current1 = list1;    // pointer
6       current2 = list2;    // pointer
7       node list3;          // head of new list
8       current3 = list3;
9       for (int I = 0; I <= 6 ;++i)
10      {
11          push(current3) ; //add new node in list
12          if (current1→rank == current2→rank )
13          {
14              currnet3->rank = current1→rank;
15              current3→coef = current1→coef + current2→coef;
16          }
17          else if (current1→rank > current2→rank)
18          {
19              currnet3->rank = current1→rank;
20              current3→coef = current1→coef;
21          }
22          else if (current1→rank < current2→rank)
23          {
24              currnet3->rank = current2→rank;
25              current3→coef = current2→coef;
26          }
27      }
28  }
29
```

Question 2:

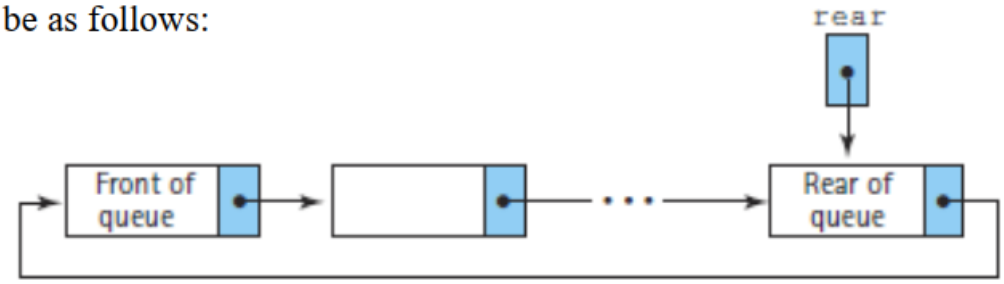
Design a data representation that maps n data structures into an array. These structures consist of n_1 stacks and n_2 circular queues such that $n_1 + n_2 = n$

a) Explain how to represent a queue with a front and a rear using a linked list in a way that guarantees efficient Deq and Enq.



b) Modify the queue given in (a) into a circular one. Use diagrams or pseudo-code.

- Using the linked list, the efficient implementation of the circular queue will be as follows:



We used only one pointer, rear.

- To Enqueue an element, we access the rear node directly through the pointer rear.
- To Dequeue an element, we access the front node of the queue. The front node is the next of the rear node.

c) Explain how can you implement such a data structure.

- we will implement both the queues and the stacks as linked lists We have n1 pointers (top) for stacks, n2 pointers (head) for linked lists, and n3 pointers (rear) for circular queues.
- top1, top3, ……, top$_{n1}$ head1, head2, ..., head$_{n2}$ rear1, rear2, …, rear$_{n3}$ Initially all the pointers equal -1 to indicate empty data structure.

- Every node will be represented as a structure with member called next holds the index of the next node in the linked list (Remember the linked list implemented in the section). When we add an item to any data structure, we access the available list to find the next empty slot in the array.
- The array is full when the available list is empty and vice versa

Question 3

You are given a sequence of **RNA** to translate it into protein (i.e sequence of amino acids) using **queue (ADT)**. Also, you are given the following functions implemented for you.

- char dna::rna2aa(char c1, char c2, char c3) that takes a codon (i.e three consecutive characters of RNA) and returns the corresponding amino acid.
- void enqueue(CharQueue &q , char data), enqueues a data into q.

```
CharQueue translateRNA( char *rna, int size )
{
    // Declare an empty queue (done)
    CharQueue amino_acids;
    // Iterate over the RNA:
    // (1) converting each 3 elements into an amino acid
    // (2) the amino-acid to be enqueued into the queue
    for (          ;          ;       )
    {


    }
    // return the queue
}
```

Also, determine the Big-Oh notation for the above function.

$$O(T(n)) =$$

```
1   charQueue translateRNA(char *rna, int size)
2       {
3           charQueue amino_acids;
4           for(int i = 0; i<size-2; i+=3)
5           {
6               Char codon = rna2aa(rna[i], rna[i+1], rna[i+2]);
7               Enqueue (amino_acids, codon);
8           }
9           Return amino_acids;
10      }
```

$$O(T(n)) = O(n/3) \text{ //because of } i+=3$$

Question 4:

Given a linked list sorted in ascending order, you are required to insert a new element, such that the list remains sorted. You may use pseudo code.

```
struct IntegerNode
{
    int data;
    IntegerNode *next;
};
struct IntegersLL
{
    IntegerNode *head;
};
void insertFront( IntegersLL &list , int data )
{


}
void insertSorted( IntegersLL &list , int data )
{


}
```

Also, determine the Big-Oh notation for the insertAt and insertSorted functions.

- insertFront: O(T(n)) =
- insertSorted: O(T(n)) =

```
1   void insertFront(IntegersLL &list, int data)
2   {
3       IntegerNode *temp = new IntegerNode{data, list.head};
4       List.head = temp;
5   }
6
7   void insertSorted(IntegersLL &list, int data)
8   {
9       IntegerNode *temp = list.head;
10      IntegerNode *aux = list.head;
11      While(((temp->data) < data )&& ((temp->next) != NULL))
12      {
13          Aux = temp;
14          Temp = temp->next;
15      }
16      IntegerNode *temp2 = new IntegerNode{data, temp};
17      Aux->next = temp2;
18  }
```

- insertFront: O(T(n)) = O(1)
- insertSorted: O(T(n)) = O(n)

Midterm 2016 Retake - Solution

Question 1:

1. Which of the following stack operations could result in stack underflow?
   a) isEmpty
   b) pop
   c) push
   d) Two or more of the above answers

2. In the linked list implementation of the stack, where does the *push* operation place the new entry on the linked list?
   a) At the head
   b) At the end of the list
   c) After all other entries that are greater than the new entry.
   d) After all other entries that are smaller than the new entry.

3. Which of the following applications may use a stack?
   a) A parentheses balancing program.
   b) Keeping track of local variables at run time.
   c) Syntax analyzer for a compiler.
   d) All of the above.

4. In the array implementation of the circular queue. If the size of the array is N what the value of the rear after *enqueue* operation.
   a) (rear % 1) + N
   b) rear % (1 + N)
   c) (rear + 1) % N
   d) rear + (1 % N)

5. In linked lists there are no NULL links in
   a) single linked list
   b) linear doubly linked list
   c) circular linked list
   d) linked list

6. Suppose *cursor* refers to a node in a linked list. What boolean expression will be true when *cursor* refers to the last node of the list?
   a) (cursor == null)
   b) (cursor.next == null)
   c) (cursor.data == null)
   d) (cursor.data == 0.0)
   e) None of the above.

Question 2:

Here is an **incorrect** pseudo-code for the algorithm which is supposed to determine whether a sequence of parentheses is balanced:

```
declare a character stack
while ( more input is available)
{
   read a character
   if ( the character is a '(' )
      push it on the stack
   else if ( the character is a ')' and the stack is not empty )
      pop a character off the stack
   else
      print "unbalanced" and exit
 }
 print "balanced"
```

Note: The parentheses are '(' and ')' only

1. What is wrong with this code?
   - (Print "balanced") is wrong we should check if the stack is empty or not so if the stack is empty, we print "balanced" if it's not empty, we print unbalanced.

2. Which of these unbalanced sequences does the above code think is **balanced**? Why?
   a) ((( ))
   b) ( )) (( )
   c) (( ) ( )))
   d) (( ))) ( )

   - The code will think "a. ((())" is balanced cause it won't get to the else statement.

3. Rewrite the above code in a correct form

```
declare a charactar stack
while ( more input is available )
{
        read a charactar
        if ( the charactar is a '(' )
                push it on the stack
        else if ( the charactar is a ')' and the stack is not empty )
                pop a charactar off the stack
        else
                print "unbalanced" and exit
}
if (stack is not empty)
        print "unbalanced"
else
        print "balanced"
```

Question 3:

Given a number of students, write a program in C++ that reads the grades of each student. then calculate the mean and the standard deviation of the grades. The standard deviation is defined as follows:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

```cpp
#include <iostream>
#include <cmath>
double square(double x)
{
    return x*x;
}
double meanArray(double *base, int size)
{
    double sum = 0;
    for(int i = 0; i<size; i++)
    {
        sum += base[i];
    }
    return sum/size;
}
double standardDeviation(double *base, int size)
{
    double sum = 0;
    double mean = meanArray(base, size);
    for(int i = 0; i<size; i++)
    {
        sum += square(base[i]-mean);
    }
    return sqrt(sum/size);
}
int main()
{
    int numOfStudents;
    std::cin >> numOfStudents;
    double grades[numOfStudents];
    for(int i = 0; i<numOfStudents; i++)
        std::cin >> grades[i];
    std::cout << "Mean = " << meanArray(&grades[0], numOfStudents) << std::endl;
    std::cout << "Standard deviation = " << standardDeviation(&grades[0], numOfStudents) << std::endl;
    return 0;
}
```

Omar
Cantera

Cairo University
Faculty of Engineering
Systems & Biomedical Eng. Dept.

Second Year
April 2016
Time: 1.15 hours

COMPUTER II

Attempt all questions

1.Assume you are working in a factory, you are asked to design and implement an inventory information system of the existing items. Each item is represented by the item name, item number, price, number on hand, supplier number(s)(the item may be ordered from different suppliers) and the country of its origin. Moreover, for each item a critical number under which a warning should be issued to order a new shipment of this item. Each supplier is described by the supplier name, supplier number, address, country, telephone, fax and e-mail.

a) What is the suitable data structure in C++to represent each item and each supplier? Implement both data structures.

b) How can you represent a list of all the suppliers?

2.A data representation that maps n data structures into an array. These structures consist of n1 stacks, n2 linked lists and n3 circular queues such that $n1 + n2 + n3 = n$

a) Using a linked list, explain how to develop a circular queue from a queue with both a Rear and Front (choose the Rear and Front locations to guarantee an efficient implementation).

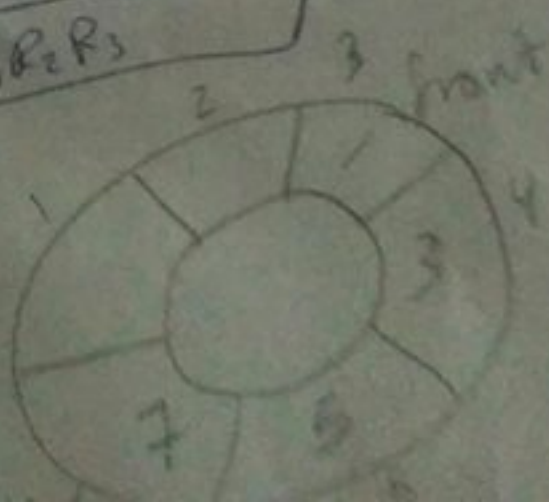
b) Explain how can you locate the empty locations in the array efficiently.

c) Explain how can you implement the data structure described above. Explain how can you check whether it is empty or full.

3. Given number of resistors ,Write a program in C++ that reads the values of of each resistor then calculate the parallel and series equivalent.

$$\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

$$\frac{R_3R_2 + R_1R_3 + R_1R_2}{R_1R_2R_3}$$



Solution of Question #1

a) struct data structure

There are many possible alternatives for the data types of the members

```
struct item
{
        char  name[20];
        int  number;
        float  price;
        int  number_on_hand;
        int  number_of_suppliers;
        char  country[10];
        int  critical_number;
};

struct supplier
{
        char  name[20];
        int  number;
        char  address[30];
        char  country[10];
        int  telephone;
        int  fax;
        char  email[20];
};
```

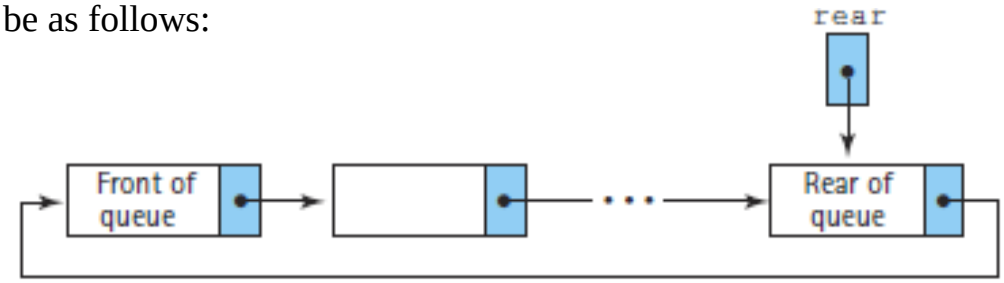
b) By using array of structures of type supplier

```
supplier  list[NUMBER_OF_SUPPLIERS];
```

Every slot in the array holds the information of one supplier.

Solution of the Question #2

a) Using the linked list, the efficient implementation of the circular queue will be as follows:



We used only one pointer, rear.

- To Enqueue an element, we access the rear node directly through the pointer rear.
- To Dequeue an element, we access the front node of the queue. The front node is the next of the rear node.

b) We can locate the empty locations in the array efficiently by using the available list.

The available list contains the indices of the empty locations in the array. Initially the list contains all the indices of the array. When we need to add an item to the array, we access the available list to find the next available space. Then we delete this location from the list. When we need to delete an item from the array, we mark this slot as empty by adding its location to the available list.

c) we will implement both the queues and the stacks as linked lists

We have n1 pointers (top) for stacks, n2 pointers (head) for linked lists, and n3 pointers (rear) for circular queues.

top1, top3, ……,top_{n1} head1, head2,.., $head_{n2}$ rear1,rear2,…,$rear_{n3}$

Initially all the pointers equal -1 to indicate empty data structure.

Every node will be represented as a structure with member called next holds the index of the next node in the linked list (Remember the linked list implemented in the section). When we add an item to any data structure we access the available list to find the next empty slot in the array.

The array is full when the available list is empty and vice versa.

Solution of Question #3

There are many possible answers to this question.

```
#include <iostream>

using namespace std;
const int NUMBER_OF_RESISTORS = 10;

int main()
{
    float resistors[NUMBER_OF_RESISTORS];
    cout << "Please Enter The Values of Resistors:" << endl;
    for(int i=0; i<NUMBER_OF_RESISTORS;i++)
    {
        cin>>resistors[i];
    }

    float series = 0;
    float parallel_inverse = 0;

    for(int i=0;i<NUMBER_OF_RESISTORS; i++)
    {
        series += resistors[i];
        parallel_inverse += (1/resistors[i]);
    }

    cout << "The Series Equivalent is: "<<series<<endl;
    cout << "The Parallel Equivalent is: "<<(1/parallel_inverse)<<endl;
    return 0;
}
```

Cairo University
Faculty of Engineering
Systems & Biomedical Eng. Dept.

Second Year
May 2016
Time: 1hour

COMPUTER II

Attempt all questions

Question NO.(1).

A storage facility uses part numbers as key fields, has the following parts with the following numbers:

n=17

23, 65, 37, 60, 16, 12, 48, 71, 62, 59, 18, 21, 10, 74, 78, 15, 14

10 12 14 15 16 18 21 23 37 48 59 60 62 65 71 74 78

a) Show how a binary search tree will expand when inserting the previous numbers and what the final tree will look like.

b) For the given values, show how you can construct a binary search tree of minimum number of levels. Is it possible to get a binary search tree of maximum number of levels? Show how?

c) What are the conditions that should be satisfied in a binary tree to construct a heap. Construct from the given numbers a heap.

d) It is required to delete the binary tree you have got in (a), What is the best tree traversal approach to accomplish this task? Show how you will delete the tree using the chosen traversal approach.

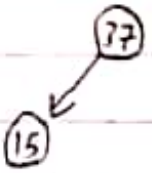
Q1 First we sort the given numbers

a)

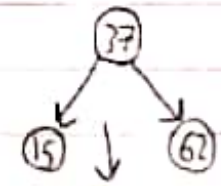
10 12 14 15 16 18 21 23 37 48 59 60 62 65 71 74 78

The root is the mid element which is 37

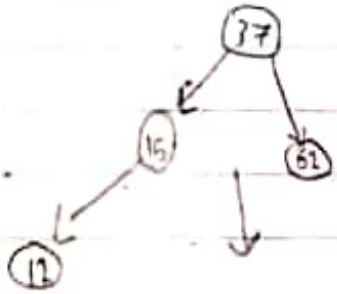
The root of the left sub tree is 15

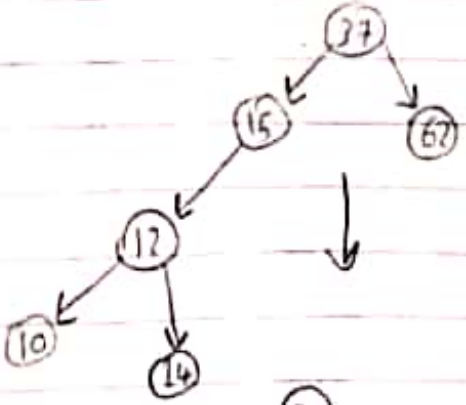


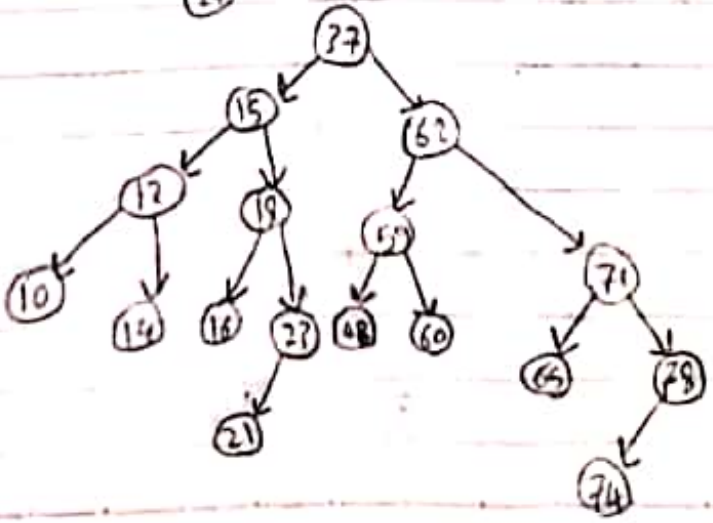
The root of the right subtree is 62



and so on.







b) minimum number of levels same as (a)

maximum number of levels

c) مش كايبا

d) we delete using bFS traversal

```
Queue q;
enqueue(q, root)
while(q is not empty)
{
    Node temp = q.front;
    dequeue(q);
    enqueue(q, temp->left);
    enqueue(q, temp->right);
    delete temp;
}
```