SULIT

SCHOOL OF COMPUTING
Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL EXAMINATION SEMESTER I, 2019 / 2020

SUBJECT CODE	**:**	**SCSJ2013**
SUBJECT NAME	**:**	**DATA STRUCTURES AND ALGORITHMS**
YEAR / COURSE	**:**	**2SCSB, 2SCSP, 2SCSJ, 2SCSR, 2 SCSV**
TIME	**:**	**3 Hours**
DATE/DAY	**:**	
VENUES	:	

INSTRUCTIONS TO THE STUDENTS:

This examination book consists of 2 parts:

Part A:	20 Objective Questions	20 marks
Part B:	4 Structured Questions	80 marks
	Question 1 (Searching)	– 15 marks
	Question 2 (Linked List)	– 15 marks
	Question 3 (Stack)	– 15 marks
	Question 4 (Queue)	– 20 marks
	Question 5 (Tree)	– 15 marks

ANSWER ALL QUESTIONS IN THE ANSWER BOOKLET PROVIDED.

Name	
Identity card / Matric Number	
Name of Lecturer	
Subject Code and Section	

This examination book consists of **14** printed pages excluding this page.

PART A – OBJECTIVE QUESTIONS **[20 marks]**

Part A consists of 20 objective questions. Each question carries 1 mark.
Choose the <u>correct answer</u> and write your answer in the test booklet.

1. Where is linear searching used?

   i. When the list has only a few elements

   ii. When performing a search in an unordered list

   iii. Used all the time

   iv. None of the above

   A. i and ii
   B. i,ii and iii
   C. ii and iii
   D. iv

2. Complexity of binary search algorithm is _____.

   A. O(n)
   B. O($\log_2 n$)
   C. O(n^2)
   D. O($n\log_2 n$)

3. The worst case occur in linear search algorithm when _____.

   i. Item is somewhere in the middle of the array

   ii. Item is not in the array at all

   iii. Item is the last element in the array

   iv. None of the above

   A. i and ii
   B. ii and iii
   C. i,ii and iii
   D. iv

4. Which of the following is NOT a limitation of binary search algorithm?

   A. must use a sorted array

   B. expensive when a lot of insertion and deletions are needed

   C. there must be a mechanism to access middle element directly

   D. binary search algorithm is not efficient when the data elements more than 2500.

5. A linear collection of data elements where the linear node is given by means of pointer is called?

   A. Linked list

   B. Node list

   C. Primitive list

   D. Pointer list

6. Which of the following operations is the main advantage of doubly linked list?

   A. Easy to traverse backwards.

   B. Searching of an unsorted list for a given item

   C. Inverting a node after the node with given location

   D. Traversing a list to process each node

7. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head and tail pointer. Given the representation, which of the following operation can be implemented in `O(1)` time?

   i) Insertion at the front of the linked list

   ii) Insertion at the end of the linked list

   iii) Deletion of the front node of the linked list

   iv) Deletion of the last node of the linked list

   A. i and ii

   B. i and iii

   C. i, ii and iii

   D. i, ii and iv

8. In linked list, each node contain minimum of two fields. One field is data field to store the data while the second field is _____.

   A. Pointer to character

   B. Pointer to integer

   C. Pointer to node

   D. Pointer to pointer

9. Which of the following is NOT an application of stack data structure?

   A. Recursive function calls in C++

   B. WhatsApp replying chats ordering

   C. Converting an infix to postfix

   D. Web browser back button

10. In linked representation of stack, the null pointer of the last node in the list indicates _____.

   A. top of the stack

   B. bottom of the stack

   C. middle of the stack

   D. in between some value

11. When a new node push onto a stack, _____ .

   A. the new node is placed at the front of the linked list

   B. the new node is placed at the back of the linked list

   C. the new node is placed at the middle of the linked list

   D. no changes happens

12. The effect of calling `stackTop( )` method in full stack array implementation is

   A. It removes the first element of an array

   B. It removes last element of an array

   C. It modifies an array start from the last element of an array

   D. None of the above

13. In a circular array based queue the value of rear (`r`) will be _____.

A. `r=r+1`
B. `r=(r+1)% [QUEUE_SIZE - 1]`
C. `r=(r+1)% QUEUE_SIZE`
D. `r=(r-1)% QUEUE_SIZE`

14. One difference between a queue and a stack is:

A. Queues require linked lists, but stacks do not.
B. Stacks require linked lists, but queues do not.
C. Queues use two ends of the structure; stacks use only one.
D. Stacks use two ends of the structure, queues use only one.

15. What should be the value of rear if the queue is full (elements are completely occupied)?

A. 1
B. - 1
C. `SIZE + 1`
D. `SIZE - 1`

16. Which of the following is NOT NEEDED in the structure of a queue implementation circular linked list?

A. Node
B. Pointer to front queue
C. Pointer to rear queue
D. Pointer **`next`** for every node in the queue

back ptr → next act as front

17. Which statement is TRUE for tree in Figure A1?

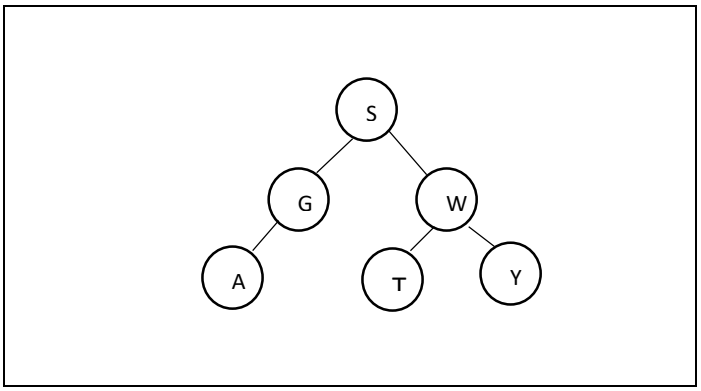


Figure A1

i. The tree is complete

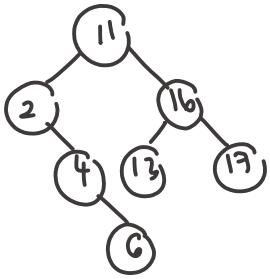
ii. The tree is full

iii. The tree is balanced

iv. The tree is binary tree

A. i and ii B. ii, iii and iv C. iii and iv D. iv only

18. The following numbers are inserted into an empty binary search tree in the given order: 11, 2, 4, 6, 16, 13, 17. What is the height of this binary search tree?



A. 5 B. 4 C.3 D.2

19. A node on the path from the root to the node *n* is called

A. Successor of node *n* B. Ancestor of node *n*

C. Internal of node *n* D. Children of node *n*

20. Match the following for binary search tree traversal

TYPE OF TRAVERSAL	ORDERING
(1) Pre Order	**(A)** Left Right Root
(2) In Order	**(B)** Left Root Right
(3) Post Order	**(C)** Root Left Right

A. 1 → C, 2 → A, 3 → B C. 1 → B, 2 → A, 3 → C

B. 1 → A, 2 → C, 3 → B D. 1 → C, 2 → B, 3 → A

PART B - STRUCTURED QUESTIONS

[80 MARKS]

Part B consists of 4 structured questions. Answer all questions in the space provided. The marks for each part of the question is as indicated.

Question 1 **[15 MARKS]**

a) Given `Program 1` and `array[]` in Figure B1, answer the following questions.

```
1   //Program 1

2   int Search(int search_key, int array[ ],int   array_size )
3   {
4     int p; int index = -1;
5     for(p = 0; p < arraysize; p++)
6      {     if ( search_key < array(p))
                 break ;
7        if ( search_key == array[p] )
8        { indeks = p; // assign current array index
9          break; //terminate searching
10       }
11
12     } //end for

13     return index; // return location of value
14  } //end function
```

3	2	4	1	5

Figure B1: `array[ ]`

i. Name the searching technique in `Program 1`. Give the reason to justify your answer. Sequential (linear) searching. (2 marks)

ii. What is the complexity time and number of comparisons when searching on `array[]` if the `search_key` is 10? O(n) . 5 number of comparison. (1 mark)

iii. Based on your answer in (ii), explain one drawback of this searching technique and provide a solution to improve it. slowest searching algorithm because using unsorted data. (2 marks)

iv. Modify `Program 1` to accomplish the suggested solution in (ii). (2 marks)

b) Given a search function in `Program 2`. Answer all the following questions based on `INPUT` array shown in Figure B2.

```
//Program 2
int search( int  search_key, int array_size, const int INPUT[] )

{   bool found = false;
    int index = -1 //-1 means record not found
      int MIDDLE, LEFT = 0,
      RIGHT = arraysize-1;

 while ((LEFT <= RIGHT ) && (!found))
 {    MIDDLE = (LEFT + RIGHT )/ 2;   // Get middle index
      if (INPUT[MIDDLE] == search_key)
      {     index = MIDDLE;
            found = true;
      }
      else if (INPUT[MIDDLE] > search_key)
            RIGHT = MIDDLE - 1;   // search is focused on the left
                                  // side of list
      else
            LEFT = MIDDLE + 1;     // search is focused on the right
                                 / / side of the list
 } //end while

 return index;
}//end function
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
5	9	19	25	34	40	45	49	66	75	88	100

Figure B2: `INPUT` array

Trace the value of **`LEFT, RIGHT, MIDDLE, INPUT[MIDDLE]`** and **`found`** (as in Table B1) for binary search operation performed onto `INPUT` array with the key numbers being search as following: (8 marks)

i. Search key = 40
ii. Search key=100
iii. Search key= 18

Table B1

LEFT	**RIGHT**	**MIDDLE**	**INPUT[MIDDLE]**	**found**

Question 2 **[15 MARKS]**

Figure B3 shows 3 nodes arranged as a single linked list, together with 5 pointer variables (`pPrev, pNext, pHead, pTail and pCurr`). Answer all the questions (a-f). Do not define any new variables, use only the variables defined in Figure B3. Each question is **independent/ not related** of each other.

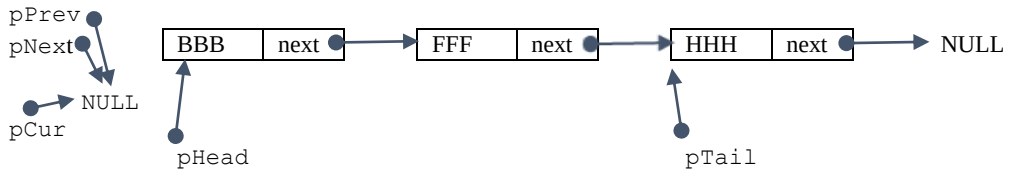


Figure B3: Single link list with 3 nodes

Answer each question (a-c) below by redrawing the Figure B3 after execution of the code statements.

a. Redraw the result to Figure B3, after the following statements execute. (2.5 marks)

```
pPrev = NULL;
pCur = pHead;
pNext = pCur->next;
pTail = pNext->next;
pHead = pCur->next;
```

b. Redraw the result to Figure B3, after the following statements execute. (3 marks)

```
pCurr = pTail;
pTail = pTail->next;
pPrev = pHead->next;
pPrev->next = NULL;
pHead->next = pCur;
```

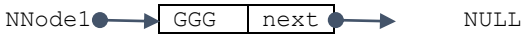
c. Redraw the result to Figure B3, after the following statements executes (2.5 marks)

```
pPrev = pNext = pHead;
while (pNext != NULL) {
      pCurr = pNext;
      pNext = pNext->next;
}//endwhile
```

Answer each question (d-f) below by writing a code segment.

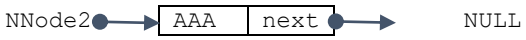
d. Given a new node, `NNode1` shown below, write a C++ code segment to insert `NNode1` in between node#2 (`FFF`) and node#3 (`HHH`) in Figure B3.

(3 marks)



e. Given a new node, `NNode2` shown below, write a C++ code segment to insert `NNode2` before node#1(`BBB`) in Figure B3.

(2 marks)



f. Write a C++ code segment to delete node#2 (`FFF`), and connect node#1 (`BBB`) and node#3(`HHH`) in Figure B3.

(2 marks)

Question 3 **[15 MARKS]**

a) Given the example in Figure B4, determine the precedence of the expression based on the precedence, associative, and parentheses rules by showing the order for each operator in the expressions. The operator that is evaluated first should be labelled as 1, the second operator to be evaluated should be labelled as 2, and so on.

$$\mathbf{x - y * z / (p - r) + s}$$ (4 marks)

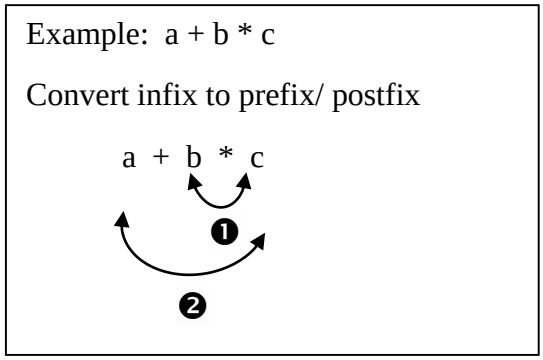


Figure B4

b) Based on your steps in a), convert the infix expression into prefix expression. (1.5 marks)

c) Based on your steps in a), convert the infix expression into postfix expression. (1.5 marks)

d) Evaluate the following postfix expression using stack concept :

```
3 9 % 20 * 7 -
```

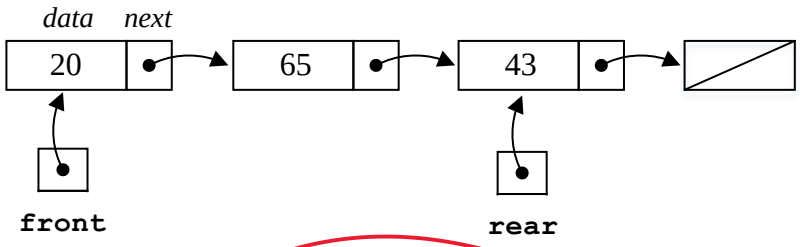
Table B2

Postfix	Ch	Op	Oprn1	Oprn2	Result	Stack
3 9 % 20 * 7 -						

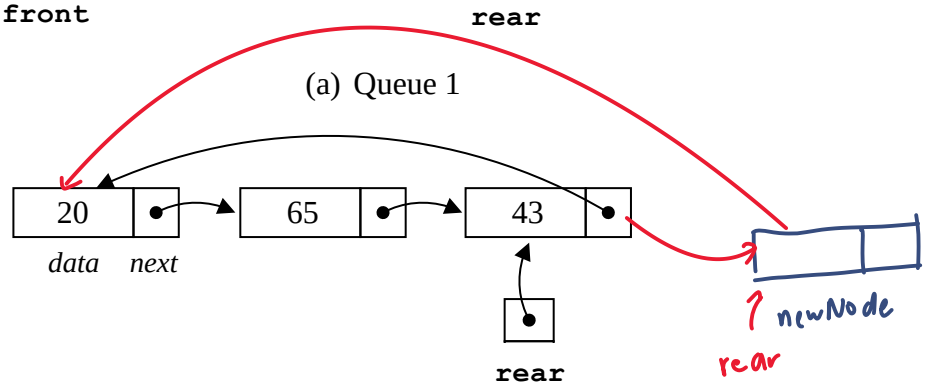
Show the evaluation process using the format given in Table B2. (8 marks)

Question 4 **[20 MARKS]**

a) Figures B5(a) and B5(b), show initial content of two different implementations of a queue.



(a) Queue 1

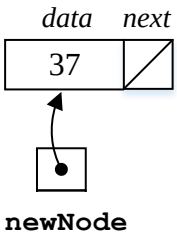


(b) Queue 2

Figure B5: Two implementations of a queue (1)

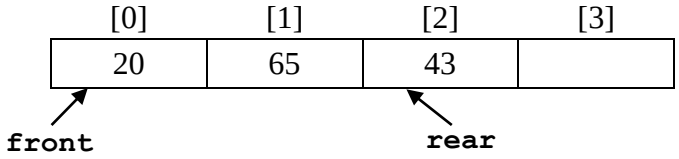
i) What are the differences between Queue 1 and Queue 2? (2 marks)

ii) Write a code segment that does insertion **`newNode`** into the Queue 1 and Queue 2. Please use the given variable name. (6 marks)

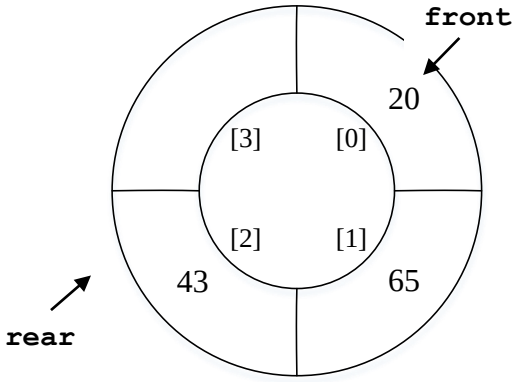


iii) Write a code segment that does deletion a node from the Queue 1 and Queue 2. Please use **`tmpNode`** to do this operation. (6 marks)

b) Figures B6(a) and B6(b), show initial content of two more different implementations of a queue. Assumed maximum size of all arrays are 4.



(a) Queue 1



(b) Queue 2

Figure B6: Two implementations of a queue (2)

Perform the following operations to the two different queues in Figure B5. Then, redraw the queues after all the operations performed. Label the correct location of the **`front`** and **`rear`**.

(4 marks)

```
enqueue(37);
dequeue();
enqueue(56);
enqueue(78);
```

Based on the answer in (a) and (b) above, which queue implementation is the best? Justify your answer.

(2 marks)

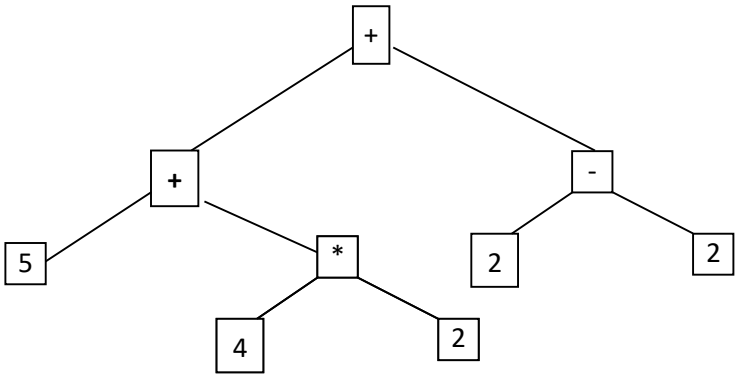
Question 5 **[15 MARKS]**

a) Based on the following infix expression, draw the expression tree that will regenerate the same infix expression when the expression tree is traverse using infix traversal

2 * 4 + 10 / 2 * (9 - 7)

(4 marks)

b) Based on the following expression tree, write the output when the expression tree is traverse using prefix and postfix.



(3 marks)

c) Based on the following sequence of numbers, draw the binary search tree (BST) based on the node insertion into the BST starting from 9 and ended with 11 in a sequential manner.

9, 6, 12, 15, 7, 10, 5, 8, 16, 11

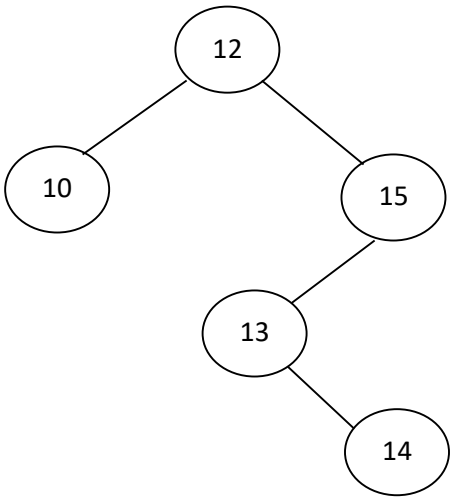
(3 marks)

d) At which level the BST in (**5-c**) is complete and full (balanced BST).

(1 mark)

e) Redraw the following tree when node 15 is deleted from the tree.

(2 marks)



f) Redraw the following tree when node 7 is inserted into the tree.

(2 marks)