SULIT

UTM UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING
Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL EXAMINATION SEMESTER I, 2018 / 2019

CODE OF SUBJECT	**: SCSJ2013**
NAME OF SUBJECT	**: Data Structures and Algorithms**
YEAR / COURSE	**: 2SCSB, 2SCSP, 2SCSJ, 2SCSR, 2 SCSV**
TIME	**: 2 Hour and 30 Minutes**

INSTRUCTIONS TO THE STUDENTS:

This examination book consists of 2 parts:

Part A:	15 Objective Questions	15 marks
Part B:	4 Structured Questions	85 marks
	Question 1 (Linked List)	– 20 marks
	Question 2 (Stack)	– 20 marks
	Question 3 (Queue)	– 25 marks
	Question 4 (Tree)	– 20 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 **12** printed pages excluding this page.

PART A – OBJECTIVE QUESTIONS **[15 marks]**

Part A consists of 15 objective questions. Each question carries 1 mark.
Choose the correct answer and write your answer in the test booklet.

1. Which of the following statements about linked list is **FALSE**?
   A. A linked list's size is virtually unlimited.
   B. A linked list requires space for pointers in each cell.
   C. Insertion / deletion of data can only be done at both head and tail of the list.
   D. A node can carry multiple data from different data types.

2. Given the following linked list in **Figure A1**, which of the following statements describe the operation to insert the **`newNode`** with value of **55** in between number **20** and **88**.

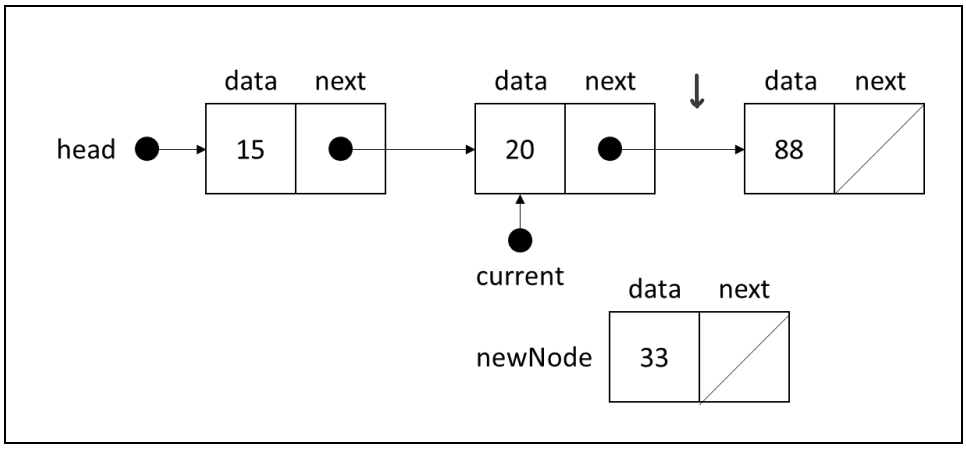


Figure A1. Singly Linked List 1

A.
```
newNode->next = current->next;
current->next = newNode;
```
B.
```
newNode->next = current;
head->next = newNode;
```
C.
```
newNode->next = current->next;
head->next = newNode;
```
D.
```
newNode->next = current;
current->next = newNode;
```

3. Which of the following applications may use a stack?
   i . A garage that is only one car wide.
   ii. Waiting in a line at Railway ticket Counter.
   iii. Compiler Syntax Analyzer.
   iv. Backtracking.

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

4. Given a linked list in **Figure A2**, which of the following statements is **INVALID**?

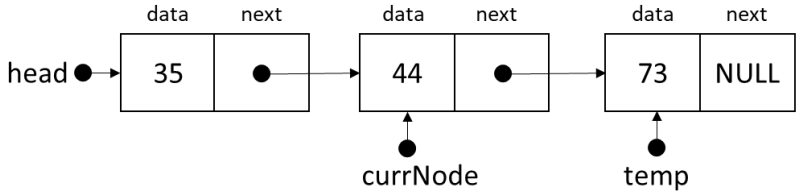


Figure A2. Singly Linked List 2

A. `temp->next = temp;`
B. `currNode->next = temp->next;`
C. `cout << currNode->next->data;`
D. `currNode = head->next->data;`

5. What function is needed to be called in stack implementation using linked list in **push** operation?
   A. `isEmpty()` function.
   B. `isFull()` function.
   C. `stackTop()` function.
   D. No function is required at all in this operation.

6. Consider the algorithm that you have learned in class for determining whether a sequence of parentheses in an expression is balanced. Give the maximum number of parentheses that appear on the stack **at the same time** when the algorithm analyses this expression:

```
(()(())(()))
```

   A. 1
   B. 2
   C. 3
   D. 4 or more.

7. The **postfix** expression for the following infix expression is :

```
A-B/(C*D)
```

   A. `A B * C D - /`
   B. `A B C D * / -`
   C. `- / * A B C D`
   D. `/ - D C * B A`

8. In the linked list implementation of linear queue, there are two pointers **`front`** and **`back`**. Which of these pointers will change during an insertion operation on a **NON-EMPTY** queue?

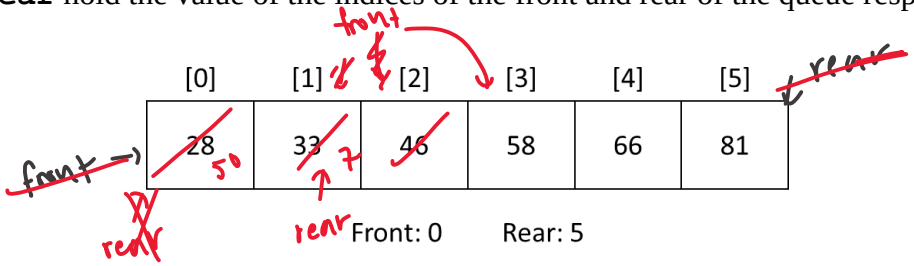
A. Only the **`back`** pointer.
B. Only the **`front`** pointer.
C. Both **`front`** and **`back`** pointer.
D. None of the mentioned above.

9. What is the purpose of the following function?

```
bool queue::func()
{
      return bool(back<front);
}
```

A. Check if the queue is full.
B. Check if the queue is empty.
C. Check the location of the pointer.
D. Retrieve the items from the queue.

10. Given the following implementation of a queue of integer values in **circular array**, **`Front`** and **`Rear`** hold the value of the indices of the front and rear of the queue respectively.



What are the **values** in the queue array at **`Front`** and **`Rear`** indices after the execution of the following queue operations?

```
dequeue();
enqueue(50);
dequeue();
dequeue();
enqueue(7);
```

A. Element at **`Front`** contains 46, and element at **`Rear`** contains 5.
B. Element at **`Front`** contains 58, and element at **`Rear`** contains 5.
C. Element at **`Front`** contains 58, and element at **`Rear`** contains 7.
D. Element at **`Front`** contains 46, and element at **`Rear`** contains 7.

11. Which of the following statements is **FALSE** about **linear array implementation** of queue?

A. **`Enqueue`** operation is not allowed to the empty spaces at the front of the array.
B. **`Enqueue`** operation is not allowed if the **`rear`** index is at the end of the array.
C. The subsequent index of **`rear`** and **`front`** is determined using arithmetic (%).
D. Only **`rear`** is used to determine whether the queue is full.

12. Based on the Binary Tree in **Figure A3**, how many **leaves** are in the tree? What is the value stored in the **parent** node of the node containing value **31**?

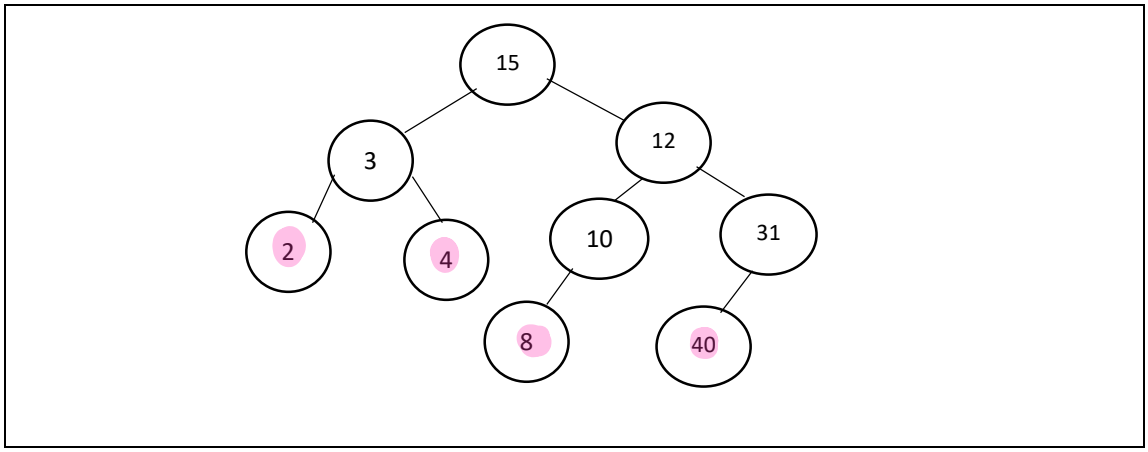


Figure A3. Binary Tree

A. The number of leaves is 6. Parent of node 31 is node 12.
B. The number of leaves is 4. Parent of node 31 is node 12.
C. The number of leaves is 6. Parent of node 31 is node 40.
D. The number of leaves is 4. Parent of node 31 is node 15.

13. Which of the following is **FALSE** about a binary search tree?

A. The left child must always lesser than its parent.
B. The right child must always greater than its parent.
C. The left and right sub-trees should also be binary search trees.
D. None of the mentioned above.

14. To obtain a **prefix** expression, which of the tree traversals is used?

A. Level-order traversal
B. Pre-order traversal
C. Post-order traversal
D. In-order traversal

15. What does the following function do?

```
void func(Tree root)
{
      func(root.left());
      func(root.right());
      cout<<root.data();
}
```

A. Pre-order traversal
B. Level order traversal
C. Post-order traversal
D. In-order traversal

PART B - STRUCTURED QUESTIONS **[85 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 **[20 MARKS]**

a) Given the linked list diagram in **Figure B1.0**, answer the questions in this section.

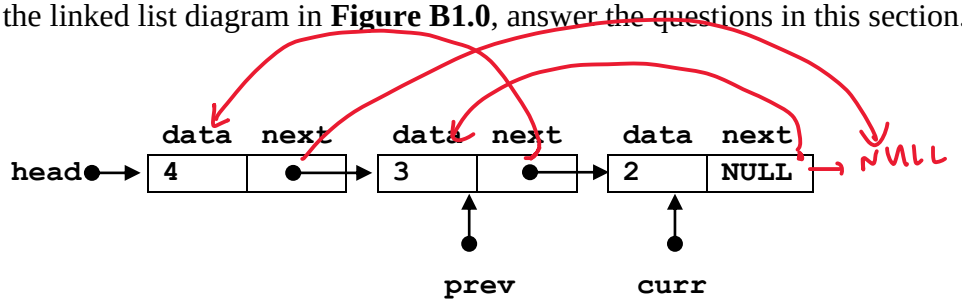


Figure B1.0. Sorted Linked List

i. Give the output if the following source codes are executed on the linked list in **Figure B1.0**.

```
temp = head;
while (temp!=NULL)
{   cout << temp->data << "->";
    temp=temp->next;
}
cout << "NULL" << end1;
```

[1.5 marks]

ii. Given the source codes below that manipulate a linked list, redraw the linked list diagram in **Figure B1.0** to show the execution of the program. Show the position of **`head`**, **`prev`** and **`curr`** in the linked list.

```
head->next = curr->next;
prev->next = head;
curr->next = prev;
```

[3 marks]

iii. Based on the new diagram drawn in Question a) ii, give the output of the program using the source codes in **Question a) i** .

[1.5 marks]

b) **`Program-1.1`** is the implementation of **`Word`** class in C++.

```
 1  // Program-1.1
 2
 3  class Word {
 4    public:
 5      string phrase;
 6      Word* next;
 7      Word* prev;
 8
 9      Word(string s) {
10        phrase = s;
11        next = NULL;
12        prev = NULL;
13      }
14  };
15
16  int main(int argc, char *argv[]) {
17
18    Word* w1 = new Word("andy");
19    Word* w2 = new Word("met");
20    Word* w3 = new Word("alice");
21    Word* w4 = new Word("yesterday");
22
23    w1->next = w2;
24    w2->next = w3;
25    w3->next = w4;
26    w4->next = w1;
27    :
28    :
29    return 0;
30  }
```

The program produces a list (circular doubly linked list) as shown in **Figure B1.1.**

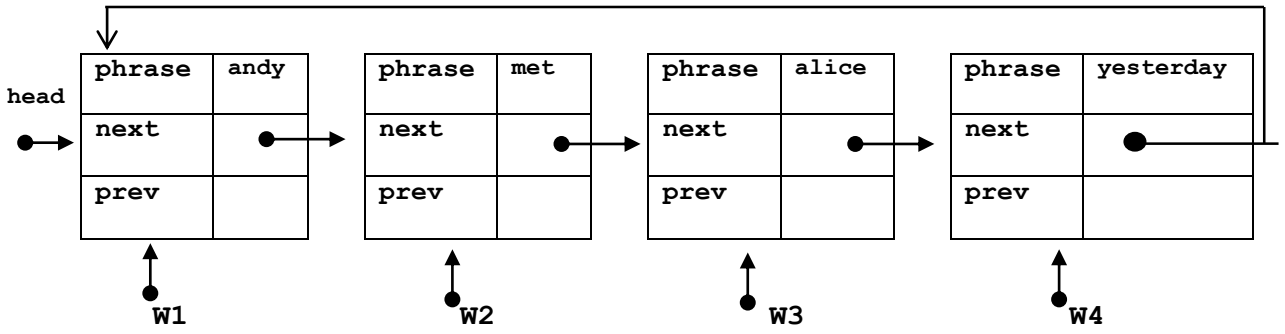


Figure B1.1 Partial Circular Doubly linked list

i. Write C++ codes to continue the implementation (after line 26) so that the above list becomes a doubly circular linked list. **Figure B1.2** shows the required structure of the list you need to produce.

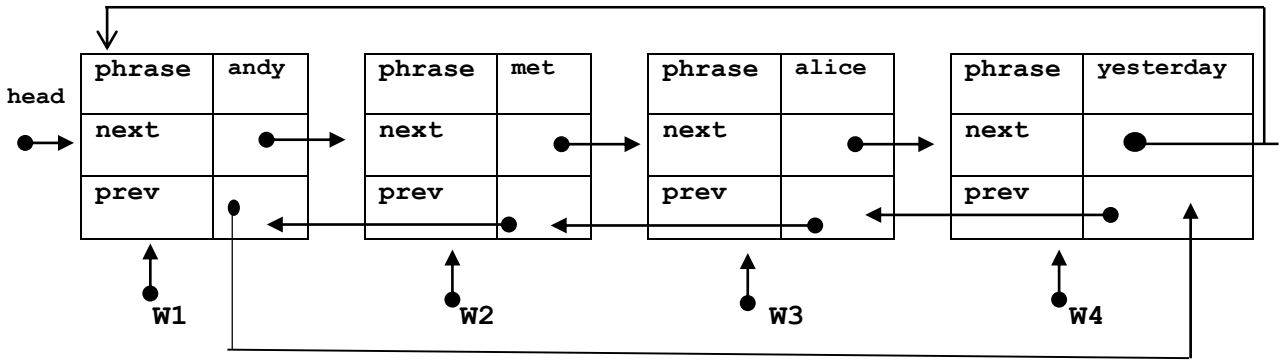


Figure B1.2. Complete Circular Doubly linked list

[4 marks]

c) Assume that the required doubly circular linked list structure of `Word` instances has been successfully constructed in Question 1 (b). Write C++ instructions to produce the following output:

```
andy met alice yesterday
yesterday alice met andy
```

You must start your answer by continuing the given C++ instructions below:

```
Word* w = w1;
while(….)
{…
 …
}
:
Word* pw = w4;
while (….)
{ …
 …
{
:
```

[10 marks]

Question 2 **[20 MARKS]**

a) Show the steps to convert the infix expression below to postfix and prefix manually.

A * B + (C % D) / (Q + R/S) – Z **[6 marks]**

b) Evaluate the following postfix expression using stack concept :

```
25 8 3 - / 6 * 10 %
```

Table B1

Postfix	Ch	Op	Oprn1	Oprn2	Result	Stack
25 8 3 - / 6 * 10 %						

Show the evaluation process using the format given in Table B1.

[8 marks]

c) Given the following stack source codes in **Figure B2.1**, show the content of the stack and the changes of **num** values, during the execution of the first **while** loop in the code segment. Use the format given in **Figure 2.2**.

[4.5 marks]

```
stack s;
int num = 13;
cout << " Decimal number : " << num << " to binary :" ;
     while((num!=0))
    {
         s.push(num%2);
         num=num/2;
     }
     while(!s.isEmpty())
     {
       cout << s.stackTop();
       s.pop();
     }
}
```

Figure B2.1. Stack implementation codes

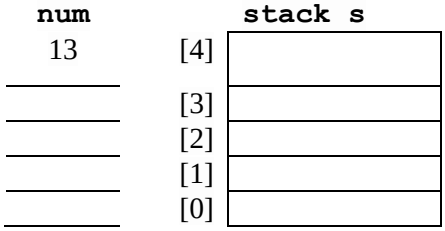
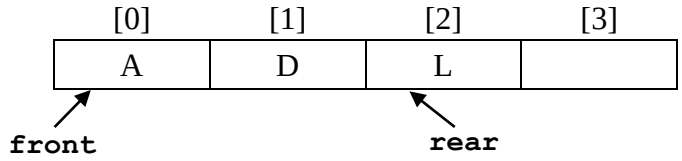


Figure 2.2 Answer Format

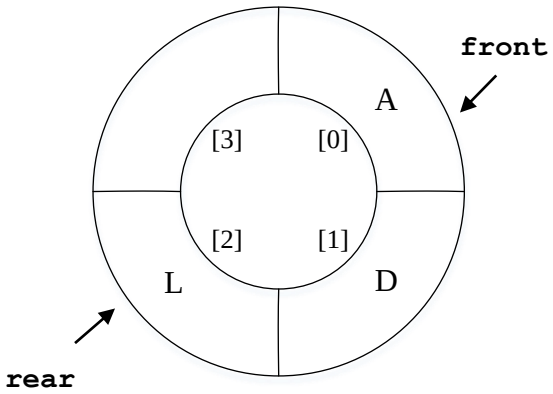
d) What is the output of the stack codes in **Figure B2.1**? **[1.5 marks]**

Question 3 **[25 MARKS]**

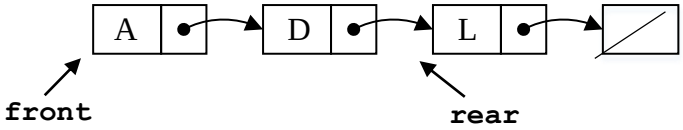
Figures B3.0 (a), (b), and (c) show initial content of three different implementations of a queue. Assume that the maximum size of all arrays are 4.



(a) Queue using linear-array



(b) Queue using circular-array



(c) Queue using singly linked-list

Figure B3.0. Three implementations of a queue

a) Give the differences between the implementation of the queue using linear-array and circular-array in the following operations: **[5 marks]**

i) Queue declaration.

ii) **`enQueue()`** function

b) Perform the following operations to the three different queues and redraw the queues for each update. Label the correct location of the **`front`** and **`rear`**. **[14 marks]**

i) `enqueue('P')`

ii) `dequeue()`

iii) `enqueue('X')`

c) Which of the queue implementation causes problem? In what operation does the problem arise? Name and describe the cause to the problem. **[4 marks]**

d) Which queue implementation is most efficient? Justify your answer. **[2 marks]**

Question 4 **[20 MARKS]**

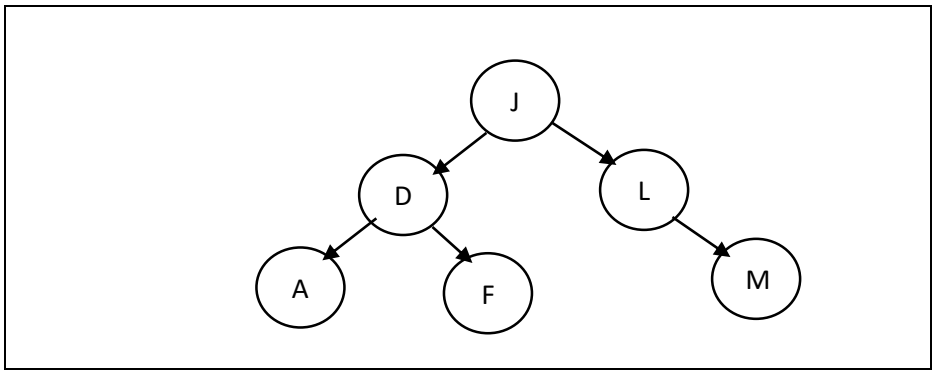


Figure B4.0

a) Based on **Figure B4.0**, answer the following questions:

i. Give the order of node visited in a in-order traversal?

ii. Give the order of node visited in a pre-order traversal?

[4 marks]

b) Consider the polynomial expression: $10y^2 - 5y$.

i. Generate an **expression tree** based on the polynomial expression by following the usual order of operations. **Hint**: to clarify the ordering for you, first write the expression with brackets indicating the order of operations.

ii. From the expression tree generated in b) i, write the polynomial as **postfix expression.**

[7 marks]

c)

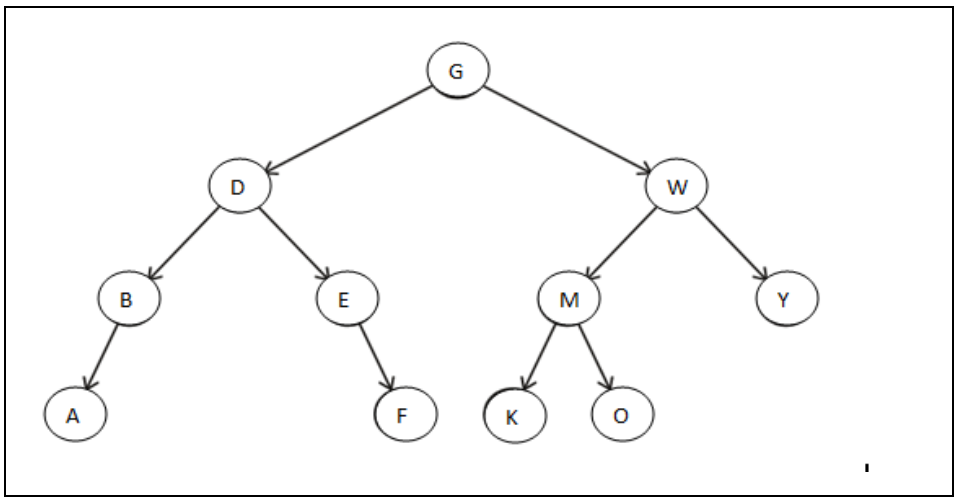
i. Draw the binary search tree based on the sequence of numbers as shown in **Figure B4.1**. The insertion starts from number 7 to 6 (left to right) in a sequential manner.

7	15	12	5	67	2	55	6

Figure B4.1. Sequence of numbers

[3 marks]

d) Given the following binary search tree, answer the following questions:



i) Height of the tree is : _____ **[1 mark]**

ii) Level of node **E** is : _____ **[1 mark]**

iii) Redraw the tree after node **G** has been deleted from the tree **[1 mark]**

iv) Referring to the original tree, redraw the tree after node **Y** and **B** has been deleted consequently **[2 mark]**

v) State whether the binary search tree drawn in iv) is full, complete or/and balanced. **[1 mark]**