SULIT

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL EXAMINATION

SEMESTER 1 2012/2013

SUBJECT CODE	**: SCSJ2013**
SUBJECT NAME	**: Data Structure and Algorithm**
YEAR / COURSE	**: 2SCV, 2SCR, 2SCJ, 2SCI, 2SCD, 2SCB**
TIME	**: (3 Hours)**
DATE	**:**
VENUE	**:**

INSTRUCTIONS TO THE STUDENTS:

This test book consists of 2 parts:

Part A:	20 Objective Questions	20 marks
Part B:	6 Structured Questions	80 marks

ANSWER ALL QUESTIONS IN THE ANSWER BOOKLET PROVIDED.

(Please write lecturer's name and section number in your answer booklet)

Name	
I/C number	
Year/Course	
Section	
Lecturer	

PART A – MULTIPLE CHOICE QUESTIONS **[20 marks]**
[*BAHAGIAN A – SOALAN PELBAGAI PILIHAN*] **[*20 markah*]**

Part A consists of 20 multiple choice questions. Choose the **CORRECT** answer and write your answer in the space given in **page 12**. Each question carries 1 mark.
[*Bahagian A mengandungi 20 soalan pelbagai pilihan. Pilih jawapan yang **BETUL** dan tulis jawapan anda pada ruang yang disediakan pada mukasurat 12. Setiap soalan bernilai 1 markah.*]

1. If an array consists of 1000 integers, what is the **maximum number of comparisons** against the target item that might be performed by binary search?
   [*Jika sesuatu tatasusunan mengandungi 1000 nombor bulat, berapakah **bilangan maksima perbandingan** terhadap item yang dicari yang perlu dilaksanakan oleh carian perduaan?*]

   A. 1
   B. 10
   C. 500
   D. 1000

2. Which of the statements below describe the **ADVANTAGE** of linked list compared to array?
   [*Yang mana diantara kenyataan-kenyataan berikut menerangkan **KELEBIHAN** senarai berpaut berbanding tatasusunan?*]

   A. Deletion a middle item will leave an empty space.
      [*Penghapusan item yang berada di tengah akan meninggalkan ruang kosong.*]
   B. A linked-list size is virtually unlimited.
      [*Saiz senarai berpaut secara maya adalah tidak terhad.*]
   C. Requires space for pointers in each cell.
      [*Memerlukan ruang untuk penuding dalam setiap sel.*]
   D. Deletion operation can be done on empty list.
      [*Operasi penghapusan boleh dilakukan ke atas senarai kosong.*]

3. What is the **worst-case** runtime complexity of searching for an item in an unsorted array using linear search?
[*Apakah masa kerumitan bagi larian **terburuk** bagi mencari item didalam tatasusunan tidak terisih dengan menggunakan carian berjujukan?*]

   A. O (n log n)
   B. O (n)
   C. O (n^2)
   D. O (n^3)

4. By referring to **Figure 1**, which of the code segments below will add a new node pointed by **`temp`** and contain value **60** at the **BEGINNING** of the list?
[*Dengan merujuk kepada **Rajah 1**, segmen kod yang manakah akan menambah satu nod baharu yang dituding oleh **`temp`** dan mengandungi nilai **60** pada **PERMULAAN** senarai?*]

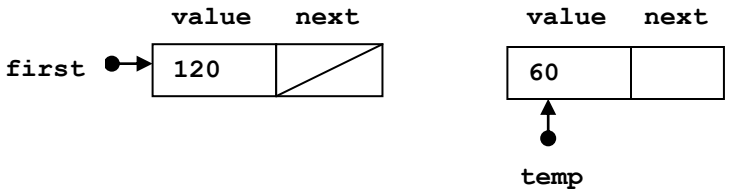


Figure 1

A. `first = new Node; first->value = 60; first->next = first;`
B. `Node* temp = new Node; first->value = 60; temp->next = first;`
C. `Node* temp = new Node;temp->value = 60;temp->next = NULL;`
   `first = temp;`
D. `Node* temp = new Node;temp->value = 60;temp->next = first;`
   `first = temp;`

5. Given the declarations below, which of the following is **TRUE** regarding **`Ptr_1`**?
[*Diberi pengistiharan berikut,* yang *manakah yang berikut adalah **BENAR** mengenai* **`Ptr_1`***?*]

```
struct ListNode
{    float     volume;
     ListNode* link;
};
ListNode *Ptr_1;
```

A. **`Ptr_1`** could be the external pointer to a linked list.
[**`Ptr_1`** *boleh menjadi penuding luar kepada senarai berpaut.*]

B. **`Ptr_1`** could point to any of the nodes in a linked list.
[**`Ptr_1`** *boleh menuding kepada sebarang nod di dalam senarai berpaut.*]

C. **`Ptr_1`** could be a node in a linked list.
[**`Ptr_1`** *adalah nod di dalam senarai berpaut.*]

D. A and B.
[*A dan B.*]

6. Which of the following is **NOT USEFUL** for the implementation of stack data structure?
[Yang manakah di antara berikut ***TIDAK BERGUNA*** *bagi perlaksanaan struktur data tindanan?*]

A. Backtracking.
[*Menjejak ke belakang.*]

B. Checking balance parenthesis.
[*Menyemak keseimbangan kurungan.*]

C. Waiting list management.
[*Pengurusan senarai menunggu.*]

D. Creating calculators
[*Menghasilkan pengira.*]

7. The following code segment **BUILDS** a linked list with the numbers 18 and 32 as its components. What should be the statement in the missing part?
[*Segmen kod berikut **MEMBINA** satu senarai berpaut dengan nilai 18 dan 32 sebagai komponennya. Apakah penyataan yang sepatutnya ada pada bahagian yang kosong?*]

```
struct NodeType
{
     int       data;
     NodeType* link;
};
NodeType* p;
NodeType* q;

    p = new NodeType;
    p->data = 18;
    q = new NodeType;
    q->data = 32;
    ……..                    // <-- Statement is missing here
    q->link = NULL;
```

A. `p = q;`
B. `p->link = new NodeType;`
C. `p->link = q;`
D. `p->link = q->link;`

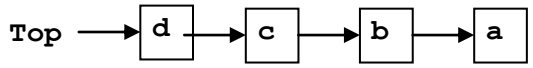
8. Which of the following codes is **CORRECT** to push a new node to a stack?
[*Yang mana diantara kod berikut adalah **BENAR** bagi menyelit satu nod ke dalam tindanan?*]

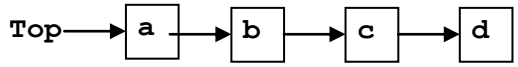
A. `Stack [Top] = NewItem; Top = Top + 1;`
B. `Top = Top + 1; Stack [Top] = NewItem;`
C. `Top = Top - 1; Stack [Top] = NewItem;`
D. `Stack [Top] = NewItem; Top = Top - 1;`

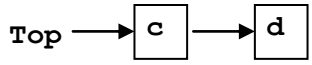
9. Given a series of stack operations as follow (assumed that initial stack is empty):
[*Diberi satu siri operasi tindanan seperti berikut (anggapkan tindanan awalan adalah kosong):*]

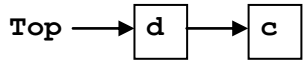
```
Push('a'), Push('b'), Pop(), Pop(),
Push('c'), Push('d')
```

What is the content of the stack after the step of operations are performed?
[*Apakah kandungan tindanan setelah langkah-langkah operasi dilaksanakan?*]

A. 

B. 

C. 

D. 

10. In evaluating the postfix expression **2 3 4 * 1 / +**, which of the following stack contents does **NOT HAPPEN** in the stack? (The last value (put in bold) is the top of the stack)
[*Dalam menilai ungkapan postfiks* **2 3 4 * 1 / +** *, yang mana diantara kandungan tindanan berikut* ***TIDAK BERLAKU*** *dalam tindanan? (Nilai akhir (yang digelapkan) adalah bahagian atas tindanan)*]

A. 2, 3, **4**
B. 2, 12, **1**
C. 2, **12**
D. 2, 3, 4, **1**

11. Which of the following expressions is the prefix notation for **A * B – D / F * G** ?
[*Yang mana di antara ungkapan berikut merupakan notasi prifiks bagi* **A * B – D / F * G***?*]

A. - * A B * / D F G
B. - * A B / D * F G
C. - * A B / * D F G
D. - * * A B / D F G

Figure 2 shows a ***circular array implementation*** of a queue data structure that holds integer values. **`front`** and **`back`** are indices that point to the head and rear elements of the queue. The array size is 5.
[*Rajah 2 menunjukkan perlaksanaan struktur data* ***baris gilir secara tatasusunan membulat*** *yang menyimpan nilai bulat. Indeks* **`front`** *dan* **`back`** *menuding kepada elemen di bahagian hadapan dan belakang baris gilir. Saiz tatasusunan adalah 5.*]

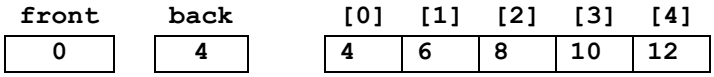


Figure 2

12. What will the ***index values*** held by **`front`** and **`back`** if the following operations are done to the queue:
[*Apakah nilai indeks bagi* **`front`** *dan* **`back`** *jika operasi-operasi berikut dilaksanakan keatas baris gilir:*]

```
dequeue()
dequeue()
enqueue(3)
dequeue()
```

A. **`front`** contains 3, and **`back`** contains 0
[**`front`** *mengndungi nilai 3 dan* **`back`** *mengndungi nilai 0*]
B. *front* contains 2, and *back* contains 4
[**`front`** *mengndungi nilai 2 dan* **`back`** *mengndungi nilai 4*]
C. *front* contains 3, and *back* contains 4
[**`front`** *mengndungi nilai 3 dan* **`back`** *mengndungi nilai 4*]
D. *front* contains 4, and *back* contains 0
[**`front`** *mengndungi nilai 4 dan* **`back`** *mengndungi nilai 0*]

Figure 3 represents the initial state of a queue data structure using **linear array** implementation. [*Rajah 3 mewakili keadaan awal bagi struktur data baris gilir yang dilaksanakan menggunakan* ***tatasusunan linear***.]

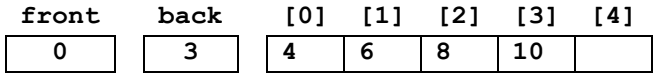


Figure 3

13. If there are two **`dequeue()`** and one **`enqueue()`** operations being implemented to the above queue, how many **`dequeue()`** operations are needed to make the queue empty? [*Jika terdapat dua operasi* **`dequeue()`** *dan satu operasi* **`enqueue()`** *dilaksanakan ke atas baris gilir, berapa operasi* **`dequeue()`** *diperlukan untuk menjadikan baris gilir kosong?*]

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

14. Which of the following statement is **FALSE** about the complete binary tree of level ***h***? [*Yang manakah diantara berikut merupakan pernyataan yang* ***SALAH*** *bagi pepohon dedua lengkap pada aras* ***h****?*]

    A. The binary tree must be full to the level ***h-1***.
    [*Pepohon dedua mesti penuh sehingga aras* ***h-1***.]
    B. At each level, the number of nodes is doubled from the previous level.
    [*Pada setiap aras, bilangan nod adalah dua kali ganda daripada aras sebelumnya.*]
    C. All nodes that are not leaf must have 2 children.
    [*Semua nod yang bukan nod daun mesti mempunyai 2 anak.*]
    D. Level ***h*** is filled from left to right.
    [*Aras* ***h*** *dipenuhi dari kiri ke kanan.*]

15. Which of the following statements describe the **CORRECT** steps in **`dequeue()`** operation in a queue linear linked list implementation?
[*Yang mana di antara kenyataan berikut, menerangkan langkah yang **BETUL** bagi operasi* **`dequeue()`** *ke atas baris gilir yang dilaksankan secara senarai berpaut linear?*]

    A. Delete the front element, and then assign the front pointer to the next element.
    [*Hapus elemen di hadapan, dan kemudian umpukkan penuding front ke elemen seterusnya.*]
    B. Assign the front pointer to the next element, and then delete the element pointed by the front pointer.
    [*Umpukkan penuding front ke elemen seterusnya, dan kemudian hapuskan elemen yang dituding oleh penuding front.*]
    C. Create a temporary pointer to point to the first element, assign the front pointer to the next element, and then delete the element pointed by the temporary pointer.
    [*Hasilkan satu penuding sementara untuk menuding ke elemen pertama, umpukkan penuding front ke elemen seterusnya, dan kemudian hapuskan elemen yang dituding oleh penuding sementara.*]
    D. Create a temporary pointer to point to the first element, assign the front pointer to NULL, and then delete the element pointed by the temporary pointer.
    [*Hasilkan satu penuding sementara untuk menuding ke elemen pertama, umpukkan penuding front kepada NULL, dan kemudian hapuskan elemen yang dituding oleh penuding sementara.*]

Figure 4 describes a circular linked list implementation of a queue data structure.
[***Rajah 4*** *menunjukkan satu struktur data baris gilir perlaksanaan senarai berpaut membulat.*]

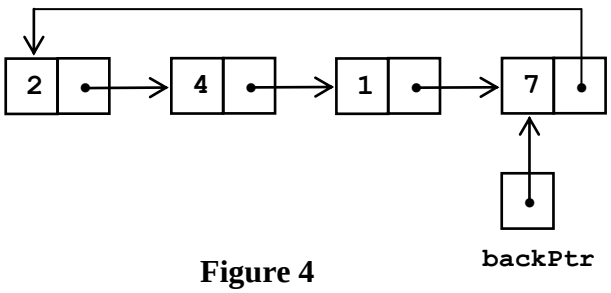


Figure 4

16. How many assignments of temporary pointer and how many changes of **`backPtr->next`** are required if we want to remove all items except the last item in the queue (the item that holds 7)?
[*Berapa bilangan umpukan bagi penuding sementara dan berapa bilangan penukaran penuding* **`backPtr->next`** *yang diperlukan sekiranya mahu menghapuskan semua item kecuali item terakhir dalam baris gilir (item yang memegang nilai 7)?*]

    A. Assignment: 3 times, changes: 3 times.
    [*Umpukan : 3 kali, penukaran : 3 kali.*]
    B. Assignment: 4 times, changes: 4 times.
    [*Umpukan : 3 kali, penukaran : 6 kali.*]
    C. Assignment: 3 times, deletion: 4 times.
    [*Umpukan : 6 kali, penukaran : 6 kali.*]
    D. Assignment: 4 times, deletion: 3 times.
    [*Umpukan : 6 kali, penukaran : 3 kali.*]

17. Every **new node** being inserted into a binary search tree will become:
[*Setiap* ***nod baru*** *yang diselit pada pepohon carian dedua akan menjadi:]*

    I. Root node to non-empty tree.
    [*Nod akar kepada sebuah pepohon yang bukan kosong.]*
    II. Leaf node.
    [*Nod daun.*]
    III. Root node to an empty tree.
    [*Nod akar kepada satu pepohon kosong*.]
    IV. Sibling to a leaf node.
    [*Adik beradik kepada satu nod daun.*]

    A. II, III
    B. I, III
    C. II, IV
    D. II, III, IV

Figure 5 shows an algebraic expression tree. Question 18-20 refer to the figure.

[***Rajah 5*** *menunjukkan satu pepohon ungkapan algebra. Soalan 17-19 merujuk kepada rajah tersebut.*]

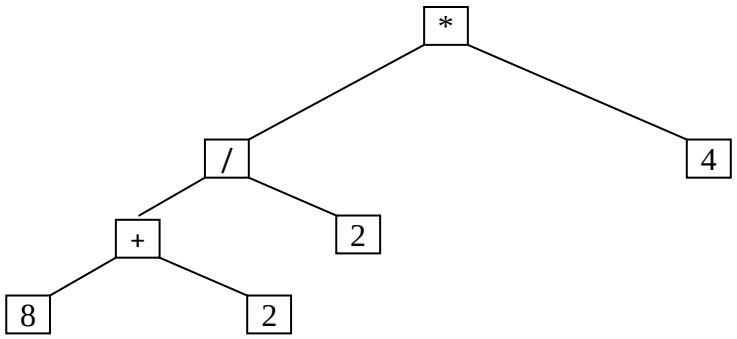


Figure 5

18. Which of the following is the **CORRECT algebraic expression** represented by the expression tree in Figure 5?
[*Yang manakah di antara berikut merupakan ungkapan algebra yang* ***BETUL*** *yang mewakili pepohon ungkapan pada Rajah 5?*]

    A. 8 + 2 / 2 * 4
    B. (8 + 2) / 2 * 4
    C. 8 + (2 / 2) * 4
    D. (8 + 2 / 2) * 4

19. Which of the following is the **CORRECT** output for **postorder** traversal of the tree in Figure 5?
[*Yang diantara berikut merupakan output yang* ***BETUL*** *untuk susuran secara* ***postorder*** *bagi pepohon pada Rajah 5?*]

    A. 8 2 + 2 / 4 *
    B. 8 + 2 / 2 * 4
    C. * / + 8 2 2 4
    D. 8 2 2 / 4 * +

20. Which of the following is the **CORRECT** result of the algebraic expression represented by the tree in Figure 5?
[*Yang manakah diantara berikut merupakan hasil yang **BETUL** bagi ungkapan algebra yang diwakili oleh peopohon ungkapan algebra pada Rajah 5?*]

    A. 13
    B. 45
    C. 1
    D. 20

Answer sheet for Part A:

Write all your answers for part A in the space below.
[Jawab semua soalan bahagian A pada ruang di bawah.]

Name: _____

Section: _____

Question No	Answer
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

Question No	Answer
11.	
12.	
13.	
14.	
15.	
16.	
17.	
18.	
19.	
20.	

PART B - STRUCTURED QUESTIONS **[80 marks]**
BAHAGIAN B – SOALAN STRUKTUR **[80 markah]**

Part B consists of **6 structured questions**. Answer all questions in the space provided. The marks for each part of the question is as indicated.
[*Bahagian B mengandungi **6 soalan struktur**. Jawab semua soalan pada ruang yang disediakan. Markah setiap soalan adalah seperti yang dinyatakan.*]

Question 1 **[10 marks]**

Given **DATA** array of even integers as in **Figure 6**, answer questions in this part.
[*Diberikan tatasusunan **DATA** bernombor bulat genap seperti dalam **Rajah 6**, jawab soalan pada bahagian ini.*]

index	[0]	[1]	…	[10]	[11]	…	[49]	[50]	[51]	…	[98]	[99]
DATA	2	4	…	22	24	…	100	102	104	…	198	200

Figure 6

a) Show each step of the search process on the **DATA** array using the given search key values.
[*Tunjukkan setiap langkah proses carian pada tatasusunan **DATA** terhadap nilai kekunci yang diberi.*]

i) Fill the table below to show the values of the variables `i`, **`LEFT`**, **`MIDDLE`**, and **`FOUND`** in order to **search <u>value 0</u>** as the search key during the searching process of **`Binary_Search()`** function.
[*Penuhkan jadual berikut untuk menunjukkan nilai-nilai pembolehubah* `i`, **`LEFT`**, **`MIDDLE`**, *dan* **`FOUND`** *dalam **mencari <u>nilai 0</u>** sebagai kekunci carian sewaktu proses carian bagi fungsi* **`Binary_Search()`**.]

[4 marks]

i	LEFT	RIGHT	MIDDLE	FOUND

ii) Fill the table below to show the values of the variables **`i`**, **`LEFT`**, **`MIDDLE`**, and **`FOUND`** in order to **search <u>value 100</u>** as the search key during the searching process of **`Binary_Search()`** function.

[*Penuhkan jadual berikut untuk menunjukkan nilai-nilai pembolehubah* **`i`**, **`LEFT`**, **`MIDDLE`**, *dan* **`FOUND`** *dalam* ***mencari <u>nilai 100</u>*** *sebagai kekunci carian sewaktu proses carian fungsi* **`Binary_Search()`**.]

[1 mark]

i	LEFT	RIGHT	MIDDLE	FOUND

b) Fill in the following table with number of steps required in searching for k**eys 0, 100 and 200**. Based on the results, discuss the efficiency of binary and linear search algorithms.

[*Penuhkan jadual berikut dengan bilangan langkah yang diperlukan untuk* ***carian kekunci 0, 100 dan 200.*** *Berasaskan kepada keputusan yang diperolehi, bincangkan keberkesanan algoritma-algoritma carian perduaan dan berjujukan.*]

[5 marks]

Search key	**Number of steps**	
	Linear	**Binary**
0		
100		
200		
Discussion		

Question 2 **[15 marks]**

The questions in this part contain Question a) and b) which are not related to each other.
[*Soalan pada bahagian ini terdiri daripada Soalan a) dan b) di mana kedua-dua soalan adalah tidak berkaian antara satu sama lain.*]

a) **Figure 7** is a singly linked list that stores the information for iPads. The list has a pointer **`head`** that points to the first node in the list.
[*Rajah 7 mengandungi satu senarai berpaut tunggal yang menyimpan maklumat iPad. Senarai tersebut dituding oleh pembolehubah penuding* **`head`** *yang menuding kepada nod pertama dalam senarai*]

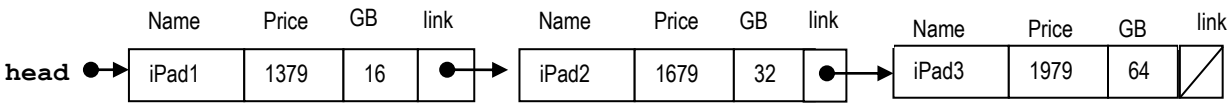


Figure 7

i) Write a **class declaration** of a node for the linked list above. Every node should have **`Name`**, **`Price`**, **`GB`** and **`link`** as the attribute. Also, define a variable called **`head`** which will point to the first node in the list.
[*Tuliskan pengistiharan nod sebagai satu kelas untuk senarai berpaut di atas. Setiap nod mempunyai medan* **`Name`**, **`Price`**, **`GB`** *dan* **`link`**. *Istiharkan juga pembolehubah* `head` *sebagai penuding kepada nod pertama dalam senarai tersebut*]

[3 marks]

ii) Write the codes statement that will display the values of all nodes in the linked list.
[*Tuliskan kod sumber yang akan memaparkan nilai-nilai kesemua nod di dalam senarai berpaut*].

[3 marks]

b) **Program 1** is the sources codes for function named **`find_location()`**. This function will find the location of node in a linear linked list which is sorted in **descending order**. In this function, the integer variable named **`item`** is the key to be searched in the list.

[*Program 1 adalah kod sumber untuk fungsi bernama* **`find_location()`**. *Fungsi ini akan mencari lokasi nod dalam senarai berpaut linear yang terisih dalam* ***turutan menurun***. *Dalam fungsi ini, pembolehubah integer bernama* **item** *adalah kunci untuk carian di dalam senarai*].

```
void List::find_location(int item)
{
  prevNode = NULL;
  currNode = head;

  while (currNode && item < currNode->data)
  {
      prevNode = currNode;
      currNode = currNode->link;
  }
}
```

Program 1

i) If the statement **`find_location(50)`** is executed and implemented on a linear linked list as shown in Figure 8, show the position of the pointer **`prevNode`** and **`currNode`** in the figure.

[*Jika penyataan* **`find_location(50)`** *dilaksanakan ke atas senarai berpaut dalam Rajah 8, tunjukkan kedudukan penuding* **`prevNode`** *dan* **`currNode`** *dalam rajah*].

[1 mark]

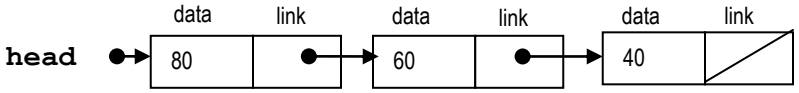


Figure 8

ii) Based on the position of the pointer variables **`head`**, **`prevNode`** and **`currNode`** drawn in part (i), write a code statement that is able to insert the **value 50** in the linked list. Use variable **`N`** which is declared and created as new node in the statement below.

[*Berdasarkan kedudukan pembolehubah* **`head`**, **`prevNode`** *dan* **`currNode`** *dalam bahagian (i), tuliskan penyataan kod yang membolehkan* ***nilai 50*** *dimasukkan ke dalam senarai berpaut. Gunakan pembolehubah* **`N`** *yang telah disitiharkan sebagai nod baru dalam penyataan di bawah*]

[3 marks]

```
Node *N=new Node;
N->data=50;
```

iii) If the statement **`find_location(20)`** is executed on a linear linked list as shown in Figure 8, show the position of the pointer **`prevNode`** and **`currNode`** in the figure.

[*Jika penyataan* **`find_location(20)`** *dilaksanakan ke atas senarai berpaut dalam Rajah 8, tunjukkan kedudukan penuding* **`prevNode`** *dan* **`currNode`** *dalam rajah*.]

[1 mark]

iv) Based on the position of the pointer variable **`prevNode`** and **`currNode`** in iii), write code segement that is able to **delete 60** from the linked list.

[*Berdasarkan kedudukan penuding* **`prevNode`** *dan* **`currNode`** *dalam iii) tuliskan segmen kod yang boleh* ***menghapus 60*** *dari senarai berpaut.*]

[3 marks]

v) What will **`head`** point after i) – iv) execution ?

[*Apakah yang dituding oleh* **`head`** *selepas perlaksanaan i) - iv)?*]

[1 mark]

Question 3 **[15 marks]**

a) Consider the following function in **Program 2** that use **`push()`**, **`pop()`** and **`isEmpty()`** operations over the stack.

[*Pertimbangkan fungsi dalam **Program 2** yang menggunakan operasi tindanan* **`push()`**, **`pop()`** *dan* **`isEmpty()`**.]

```
int final()
{    int a[8], i=0, j, b;
     while ( !isEmpty(stack) )
     {
         a[i] = pop(stack);
         if (i == 0)
              b = a[i];
         else if (a[i] > b)
              b = a[i];
         i++;
     }
     for (j=i-1; j>=0; j--)
         push(stack,a[j]);
     return b;
}
```

Program 2

i) Fill in the table below to show step by step the execution of the function in Program 2 by considering the values in the stack, the array, a and value b.

Note : Element underlined and bold is top of stack.

[*Isikan jadual di bawah dengan menunjukkan langkah demi langkah pelaksanaan fungsi dlm Atur Cara 2 dengan mengingati nilai dalam tindanan, array, a dan b.*]
[*Nota: Elemen yang bergaris dan digelapkan adalah bahagian atas tindanan.*]

stack	a[]	b
5 | 17 | 12 | 34 | 72 | **25** | |		
Content of stack and a[] array, after the execution of the for loop		

[7 marks]

ii) Describe the return value of the function. What does the function do?
[*Terangkan nilai yang dipulangkan oleh fungsi tersebut. Apakah yang dilakukan oleh fungsi tersebut?*]

[2 marks]

b) In the table given as follows, show step-by-step conversion of the expression:

A * (B – C / D) + E * F

to its postfix notation using stack operations.

[*Dalam jadual yang diberi berikut, tunjuk langkah demi langkah penukaran ungkapan:*

A * (B – C / D) + E * F

kepada notasi postfiknya dengan menggunakan operasi-operasi tindanan.]

Infix Char	**Stack**	**Postfix**

[6 marks]

Question 4 **[15 marks]**

a) Explain the problem that occurs in queue implementation linear array that can be solved by using queue circular array.
[*Terangkan masalah yang berlaku dalam pelaksanaan baris gilir secara tatasusunan linear yang boleh diselesaikan dengan menggunakan baris gilir secara tatasusnan membulat.]*

[2 marks]

b). Give **two** advantages of implementing a queue using linked list.
[*Berikan **dua** kebaikan perlaksanaan baris gisilr menggunakan senarai berpaut.*]

[2 marks]

c). **Program 3** describes class Queue declaration using circular linked list:
[*Atur cara 3 menerangkan pengishtiharan kelas Queue menggunakan senarai berpaut membulat.*]

```
struct Node {
  char item;
  Node *next;
};
class Queue {
  private:
    Node *backPtr;    //pointer to the rear item
  public:
    Queue ();         //constructor
    ~Queue();         //destructor

    bool isEmpty();

    void enqueue(char);    //do enqueue operation
    void dequeue();   //do dequeue operation

    char getFront();  //return the front item
    char getRear();   //return the rear item
};
```

Program 3

Based on the above program, answer the following questions.
[*Berdasarkan atur cara di atas, jawab soalan berikut.*]

i) To what value **`backPtr`** variable should be assigned when constructor of **`Queue`** class is called?
[*Kepada nilai apakah pemboleh ubah* **`backPtr`** *perlu diumpuk apabila constructor kelas* **`Queue`** *dipanggil.*] [1 mark]

ii) How to access the rear item using **`backPtr`**?
[*Bagaimana mencapai item dibelakang dengan menggunakan* **`backPtr`***?*] [1 mark]

iii) How to access the front item using **`backPtr`**?
[*Bagaimana mencapai item dihadapan menggunakan* **`backPtr`***?*] [1 mark]

iv) Write code segments or describe the steps to do **`enqueue()`** operation when the queue is empty.
[*Tuliskan segemen kod atau terangkan langkah-langkah untuk melaksanakan operasi* **`enqueue()`** *apabila baris gilir adalah kosong.*]
[4 marks]

v) Write code segments or describe the steps to do **`dequeue()`** operation when the queue contains more than one item.
[*Tuliskan segemen kod atau terangkan langkah-langkah untuk melaksanakan operasi* **`dequeue()`** *apabila baris gilir mengandungi lebih dari satu item.*]
[4 marks]

Question 5 **[15 marks]**

8	3	11	1	5	9	10	14	7	15

Figure 9: Sequence of numbers

a. Draw the binary search tree based on the sequence of numbers as shown in Figure 9. The insertion starts from number 8 to 15 (left to right) in a sequential manner.
[*Lukiskan pepohon carian dedua berdasarkan kepada turutan nombor dalam Rajah 9. Selitan nombor ke dalam pepohon carian dedua bermula dengan nilai 8 hingga 15 (dari kiri ke kanan) secara berjujukan.*]

[4 marks]

Question **(1-b) to (1-d)** will **ALL** be based on the binary search tree in Figure 10.

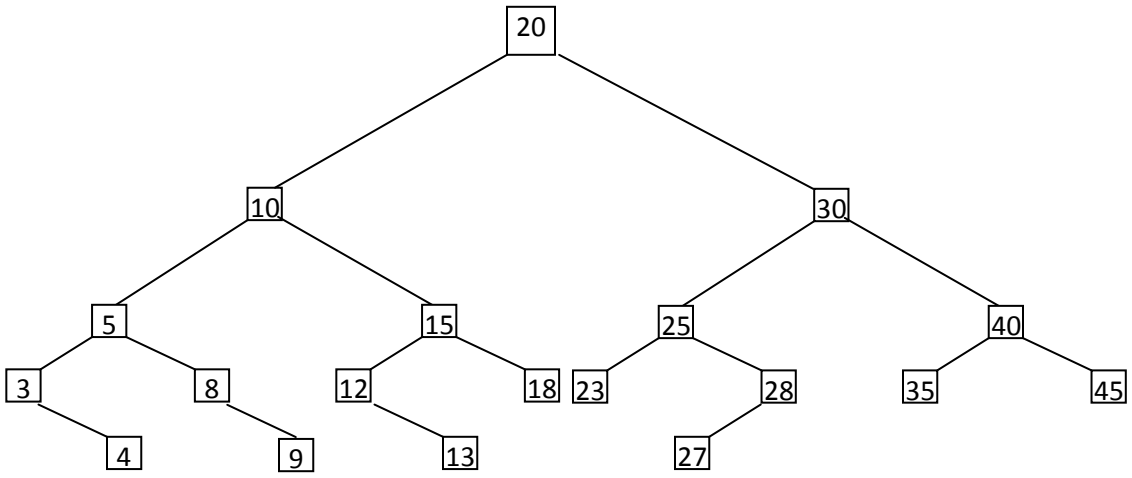


Figure 10: Binary Search Tree

b. At which level the binary search tree is **full**?
[*Pada aras manakah pepohon carian dedua tersebut **penuh**?*]

[2 marks]

c. In **Figure 11,** add 3 more new nodes with the value of your choice so that the binary search tree will become a **complete** binary search tree.
[*Pada Rajah 11, tambah 3 lagi nod baru dengan nilai yang anda tentukan sendiri supaya pepohon carian dedua tersebut menjadi pepohon carian dedua* ***lengkap****.*]

[3 marks]

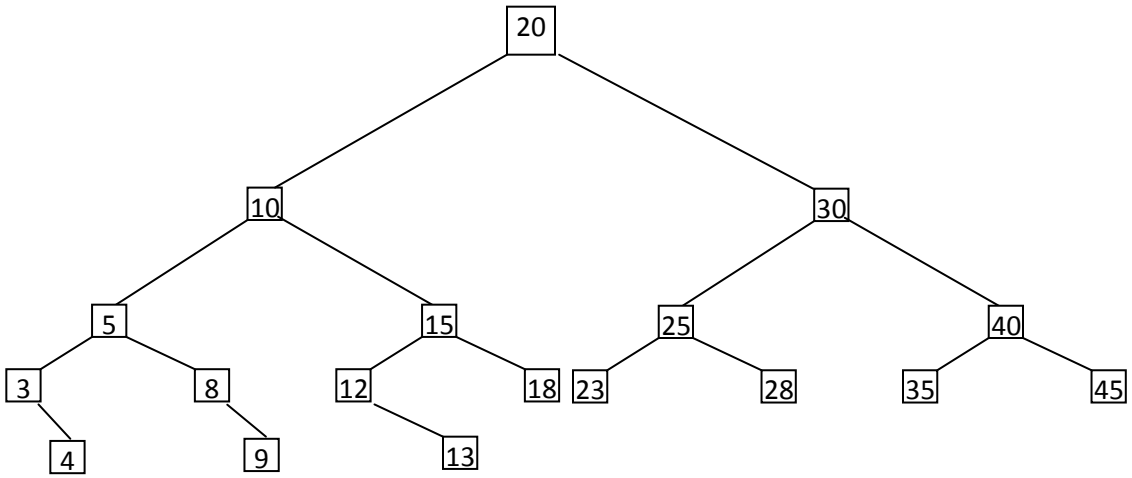


Figure 11

d. Redraw the binary search tree in Figure 10 if **node 20 is removed**.
[*Lukiskan semula pepohon carian dedua dalam Rajah 10 jika* ***nod 20 dihapuskan***.]

[3 marks]

e. Redraw the binary search tree in Figure 10 if **node 10 is removed**.
[*Lukiskan semula pepohon carian dedua dalam Rajah 10 jika* ***nod 10 di buang***.]

[3 marks]

Question 6 **[10 marks]**

Hand phone has become a necessity in our daily life. We use it for communication and to send message or emails. Among the important feature of hand phone is to store contact list which contain information about the contact. User can call, send message and e-mail to the contact in the list.

[*Telefon bimbit telah menjadi satu keperluan dalam kehidupan harian kita. Ia digunakan untuk berkomunikasi dan untuk menghantar mesej atau e-mel. Di antara ciri penting telefon bimbit ialah untuk menyimpan senarai kenalan yang mengandungi maklumat tentang kenalan. Pengguna boleh menalipon dan menghantar mesej kepada kenalan di dalam senarai kenalan.*]

Given the following situation, identify the **most suitable data structure** that can be implemented in each situation.

[*Diberikan situasi berikut, kenal pasti* ***struktur data yang paling sesuai*** *yang boleh dilaksanakan dalam setiap situasi.*]

1) From the contact list, you can view the contact in sorted order based on the contact name or search contact based on certain alphabet. You can also add, delete or edit contact at any point of the list. After each operation, the list is still in sorted order. Identify whether **array or linked list** will become the most suitable data structure that able to store the contact information efficiently for easy retrieval, add and delete. Please justify, why you choose the data structure.

   [*Dari senarai kenalan, anda boleh melihat senarai kenalan dalam keadaan terisih berdasarkan nama kenalan atau melakukan carian kenalan berdasarkan abjad tertentu. Anda juga boleh menambah, menghapus atau mengedit kenalan pada mana-mana bahagian senarai. Selepas setiap operasi, senarai masih dalam keadaan terisih. Kenal pasti sama ada tatasusunan atau senarai berpaut akan menjadi struktur data yang paling sesuai bagi menyimpan maklumat kenalan dengan cekap dan memudahkan operasi capaian semula maklumat, tambah dan hapus. Terangkan mengapa anda memilih struktur data tersebut.*]

   [4 marks]

2) Identify the **most suitable** data structure that able to store the messages received, with the most recent message at the top of the list. Explain how it is implemented.

   [*Kenalpasti struktur data yang **paling sesuai** yang dapat menyimpan mesej yang diterima, dengan mesej yang paling terkini berada di bahagian atas senarai. Terangkan bagaimana ia dilaksanakan.*]

   [3 marks]

3) Explain whether binary search tree can be used to store the contact list. If your answer is YES, give one advantage of storing the contact list in binary search tree and describe how binary search tree can be used to store the contact list. If your answer is NO, give one disadvantage of using binary search tree to store the contact list and suggest how the contact list should be stored efficiently.

   [*Terangkan sama ada pepohon carian dedua boleh digunakan untuk menyimpan senarai kenalan tersebut. Jika jawapan anda adalah ya, beri satu kelebihan menyimpan senarai kenalan dengan menggunakan pepohon carian dedua dan terangkan bagaimana pepohon carian dedua boleh digunakan untuk menyimpan senarai kenalan. Jika jawapan anda adalah tidak, beri satu kelemahan penggunaan pepohon carian dedua dalam menyimpan senarai kenalan dan cadangkan bagaimana senarai kenalan harus disimpan dengan lebih cekap.*]

   [3 marks]