UNIVERSITI MALAYSIA PERLIS

Peperiksaan Semester Pertama
Sidang Akademik 2012/2013

12 Januari 2013

EKT 334 – Algorithm and Data Structure
Algoritma dan Struktur Data

Masa: 3 jam

Please make sure that this question paper has **TWELVE (12)** printed pages including this front page before you start the examination.
*(Sila pastikan kertas soalan ini mengandungi **DUA BELAS (12)** muka surat yang bercetak termasuk muka hadapan sebelum anda memulakan peperiksaan ini.)*

This question paper has **SIX (6)** questions. **Answer FIVE (5)** questions only. Answer **ALL** questions from **Section A** and **ONE (1)** question from **Section B**. Each question contributes 20 marks.
*[Kertas soalan ini mengandungi **ENAM(6)** soalan. Jawab **LIMA (5)** soalan sahaja. Jawab **SEMUA** soalan dari **Bahagian A** dan **Satu (1)** soalan dari **Bahagian B.** Markah bagi tiap-tiap soalan adalah 20 markah.)*

Dicetak oleh Unit Peperiksaan & Pengijazahan, Bahagian Pengurusan Akademik, Jabatan Pendaftar.

SECTION A

[BAHAGIAN A]

QUESTION 1

[SOALAN 1]

a) Contrast between best, worst and average case complexities of an algorithm.

[Kontraskan di antara kes terbaik, terburuk dan pertengahan kerumitan algoritma.]

[3 Marks/Markah]

b) Consider $T_A(n)$, $T_B(n)$ and $T_C(n)$ are the time complexities of three program fragments A, B and C where $T_A(n) = O(f(n))$, $T_B(n) = O(g(n))$ and $T_C(n) = O(h(n))$. Since $T_A(n) \leq a . f(n)$ for some positive integers a and n_A such that $n \geq n_A$, $T_B(n) \leq b . g(n)$ for some positive integers b and n_B such that $n \geq n_B$ and $T_C(n) \leq c . h(n)$ for some positive integers c and n_C such that $n \geq n_c$ .Compute:

[Pertimbangkan T_A (n), T_B (n) dan T_C (n) adalah kerumitan masa tiga serpihan program A, B dan C di mana T_A (n) = O (f (n)), T_B (n) = O(g (n)) dan T_C (n) = O (h (n)). Sejak T_A (n) $\leq$ a. f (n) bagi beberapa integer positif a dan n_A seperti yang $n \geq n_A$, T_B (n) $\leq$ b. g (n) bagi beberapa integer positif b dan N_B seperti yang $n \geq n_B$ dan T_C (n) $\leq$ c. h (n) untuk beberapa integer positif c dan n_C seperti yang $n \geq n_c$ Kira:]

i) $T_A(n) + T_B(n) + T_C(n)$

ii) $T_A(n) . T_B(n) . T_C(n)$

[3 Marks/Markah]

c) Analyze the behaviour of the following program which computes the Fibonacci number, for appropriate values of ***n***. Calculate the frequency of the statements in *Table 1.1* (that are given line numbers) for various cases of **n.**

[Analisa tingkah laku program berikut yang mengira nombor Fibonacci, untuk nilai-nilai yang sesuai n. Kirakan kekerapan kenyataan dalam Jadual 1.1 (yang diberi nombor barisan) untuk pelbagai kes n.]

[7 Marks/Markah]

Table 1.1

[Jadual 1.1]

Line No	Statements
	procedure Fibonacci ***(n)***
1	**read** (n) ;
2	**If *(n*** < 0) **Then {**
3,4	**print**("error"); **exit(); }**
5	**If** (n=0) **Then**
6,7	**{ print**("Fibonacci number is 0); **exit(); }**
8	**If** (n=1) **Then**
9,10	**{ print**("Fibonacci number is 1); **exit(); }**
11,12	$f_1 = 0$; $f_2 = 1$;
13.	**for** i = 2 **to** n **do**
14,15,16	$f = f_1 + f_2$; $f_1 = f_2$; $f_1 = f$;
17.	**End**
18.	**print** ("Fibonacci number is", f*);*
	end Fibonacci

d) *Figure 1.1* shows the recursive algorithm for Tower of Hanoi puzzle. The recurrence relation for this algorithm is derived as follows: Let T(N) be the minimum number of transfers that are needed to solve the puzzle with N disks. From the function TRANSFER it is evident that for N = 0, no disks are transferred. Again for N > 0, two recursive calls each enabling the transfer of (N - 1) disks and a single transfer of the last (largest) disk from peg *S* to peg D are done.

[Rajah 1.1 menunjukkan algoritma rekursi untuk Menara teka-teki Hanoi. Perkaitan berulang bagi algoritma ini diperolehi seperti berikut: Mari T (N) menjadi bilangan minimum pemindahan yang diperlukan untuk menyelesaikan teka-teki dengan cakera N. Dari fungsi PEMINDAHAN ia adalah jelas bahawa untuk N = 0, tiada cakera dipindahkan. Lagi untuk N> 0, dua panggilan rekursi masing-masing membolehkan pemindahan (N - 1) cakera dan pemindahan tunggal cakera terakhir (terbesar) dari tambatan S untuk tambatan D dilakukan.]

```
Procedure TRANSFER(N, S, I, D)
/* N disks are to be transferred  from peg S to peg D with peg I as the intermediate peg*/
If  (N = 0) Then exit();
Else
{
  TRANSFER(N-1, S, D, I ) ;
   /* transfer N-1 disks from peg S to peg I with peg D as the intermediate peg*/
   Transfer disk from S to D;
   /* move the disk which is the last and the largest disk, from peg S to peg D*/
   TRANSFER (N-1, I, S, D) ;
  /* Transfer N-1 disks from peg I to peg D with peg S as the intermediate peg*/
End TRANSFER.
```

Figure 1.1
[Rajah 1.1]

i) Generate recurrence relation for the algorithm in *Figure 1.1*.
[Janakan perkaitan berulang untuk algoritma dalam Rajah 1.1.]

[2 Marks/Markah]

ii) Compute the time complexity from the recurrence relation.
[Kirakan kerumitan masa dari perkaitan berulang.]

[5 Marks/Markah]

QUESTION 2
[SOALAN 2]

a) Compare the advantages and disadvantages of Single Arrays over Singly Linked Lists.

[Bandingkan kebaikan dan keburukan tatasusunan tunggal berbanding Senarai Pautan Tunggal.]

[4 Marks/Markah]

b) A programming language permits indexing of arrays with character subscripts; for example, **CHR_ARRAY ['A': 'D']**. In such a case the elements of the array are **CHR_ARRAY['A'], CHR_ARRAY['B']** etc. and the *ordinal number (ORD)* of the characters viz., **ORD('A')** = 1, **ORD('B')** = 2, **ORD('Z')** = 26 and so on are used to denote the index.

Suppose three 2-Dimentional arrays **INT [1: 5, 1: 4]** (*size of the memory location: 2 bytes*), **CHAR ['A' : 'Z',1: 3]** (*size of the memory location: 1 byte*) and **REAL[1:3, 1:3]** (*size of the memory location: 4 bytes*) are stored in the memory sequentially and beginning from address **200.** Calculate the address for the following terms:

[Satu bahasa pengaturcaraan membenarkan Pengindeksan tatasusunan dengan subskrip aksara, sebagai contoh, CHR_ARRAY ['A': 'D']. Dalam kes sedemikian elemen array CHR_ARRAY ['A'], CHR_ARRAY ['B'] dan lain-lain dan nombor ordinal (ORD) aksara, ORD ('A') = 1, ORD ('B ') = 2, ORD (' Z ') = 26 dan sebagainya digunakan untuk menunjukkan indeks.

Katakan tiga tatasusunan 2-Dimensi INT [1: 5, 1: 4] (saiz lokasi ingatan: 2 bait), CHAR ['A': 'Z', 1: 3] (saiz lokasi ingatan: 1 bait) dan REAL [01:03, 1:03] (saiz lokasi ingatan: 4 bait) disimpan dalam memori berturutan dan bermula alamat dari 200. Kira alamat bagi terma berikut:]

(i) INT [4, 2]

[3 Marks/Markah]

(ii) Base address of the **CHAR** array
[Alamat Base array CHAR]

[3 Marks/Markah]

(iii) CHAR['P',2]

[3 Marks/Markah]

(iv) Base address of the **REAL** array
[Alamat Base array REAL]

[3 Marks/Markah]

6/12...

c) Consider the singly linked list **L** shown in *Figure 2.1(a)*. A new node contains a string "*Where_am_I*" need to add with the **L** linked list as depicted in *Figure 2.1(b)*. Design an algorithm which would do the operation for *Figure 2.1(b)*.

[Pertimbangkan senarai secara tunggal pautan L ditunjukkan dalam Rajah 2.1 (a). Satu nod baru mengandungi rentetan "Where_am_I" perlu untuk ditambah dengan senarai pautan L seperti yang digambarkan dalam Rajah 2.1 (b). Reka bentuk algoritma yang akan melakukan operasi bagi Rajah 2.1 (b).]

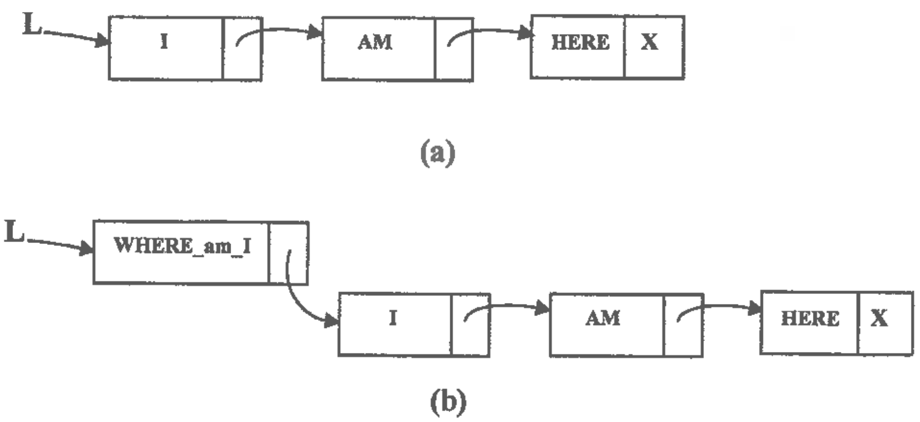

Figure 2.1
[Rajah 2.1]

[4 Marks/Markah]

QUESTION 3
[SOALAN 3]

a) Identify the demerits of linear STACK and linear QUEUE.

[Kenal pasti kelemahan TIMBUNAN linear dan BERBARIS linear.]

[4 Marks/Markah]

b) Assume a Stack **S[1:n]** algorithm (in *Figure 3.1(a) and (b)*) were to be implemented with the bottom of the stack at S[n]. Design algorithm to undertake PUSH and POP for this new operation on S.

[Andaikan algoritma S Stack [1: n] (dalam Rajah 3.1 (a) dan (b)) untuk dilaksanakan dengan bahagian bawah timbunan di S [n]. Reka bentuk algoritma untuk melaksanakan PUSH dan POP untuk operasi ini baru di S.]

[6 Marks/Markah]

```
Implementation of PUSH
operation on a stack S.

procedure PUSH (S, n, top, item)
if (top = n) then STACK_FULL;
else
{
   top = top + 1 ;
   S[top] = item;
}
End PUSH
```

(a)

```
Implementation of POP operation
on a stack

procedure POP (S, top, item)
if (top = 0) then STACK_EMPTY;
else
{
  item = S[top];
  top = top - 1 ;
}
End POP
```

(b)

Figure 3.1
[Rajah 3.1]

c) Consider **Q[1:4]** and **S[1:3]** are linearly defined Queue and Stack respectively. Assume that initially the Queue and Stacks are empty. Given A, B and C be integer variables. Demonstrate the output of the segment of pseudo code in *Table 3.1*. Following are some functions used in the Queue and Stack operation:

[Pertimbangkan Q [1:04] dan S [1:03] iaitu ditakrifkan sebagai Queue dan Stack linear masing-masing. Anggapkan bahawa mulanya Queue dan Stack adalah kosong. Diberikan A, B dan C sebagai pembolehubah integer. Tunjukkan output segmen kod pseudo dalam Jadual 3.1. Berikut adalah beberapa fungsi yang digunakan dalam operasi Queue dan operasi Stack:]

ENQ(Q, ITEM) : inserts an **ITEM** into **Q**
[memasukkan ITEM ke Q]
DEQ(Q, ITEM) : deletes an element from **Q** through **ITEM.**
[memadam elemen dari Q melalui ITEM.]

EMPTY_Q (Q) : a Boolean function which returns true if **Q** is empty and false otherwise.
[fungsi Boolean yang mengembalikan benar jika Q adalah kosong dan palsu sebaliknya.]

PRINT(ITEM) : displays the value of **ITEM.**
[memaparkan nilai ITEM.]

PUSH(S, ITEM) : inserts an **ITEM** into **S**
[memasukkan ITEM ke S]

POP(S, ITEM) : deletes an element from **S** through **ITEM.**
[memadam elemen dari S melalui ITEM.]

EMPTY_S (S) : a Boolean function which returns true if **Q** is empty and false otherwise.
[fungsi Boolean yang mengembalikan benar jika Q adalah kosong dan palsu sebaliknya.]

[10 Marks/Markah]

Table 3.1
[Jadual 3.1]

Line No.	
1	A=3
2	B=4
3	C=A+B
4	**While** (C < 150) **do**
5	**If** (C % 2) = 0 **Then** PUSH(S,C)
6	**Else** ENQ(Q,C)
7	A = B;
8	B = C
9	C = A + B
10	**End While**
11	**While not** EMPTY_S (S) **do**
12	POP(S, C)
13	PRINT (C)
14	**End While**
15	**While not** EMPTY_Q (Q) **do**
16	DEQ(Q, C)
17	PRINT (C)
18	**End While**
19	**End**

Answer 3(c) Table Hints:

Steps	S[1:4]			Q[1:4]				Variables						PRINT(C)
	[1]	[2]	[3]	[1]	[2]	[3]	[4]	Top	Front	Rear	A	B	C	
1,2,3								0	0	0	3	4	7	

Dicetak oleh Unit Peperiksaan & Pengijazahan, Bahagian Pengurusan Akademik, Jabatan Pendaftar.

QUESTION 4
[SOALAN 4]

a) Prove that for a non-empty binary tree T if n_0 is the number of leaf nodes, n_2 the number of nodes of degree 2 then $n_0 = n_2 + 1$.

[Buktikan untuk pokok binari T bukan-kosong jika n_0 adalah bilangan nod daun, n_2 bilangan nod darjah 2 maka $n_0 = n_2 + 1$.]

[4 Marks/Markah]

b) A binary tree T has 9 nodes. Design the binary tree T. The inorder and preorder traversals of T yield the following:

[Pokok binari T mempunyai 9 nod. Rekabentuk pokok binari T. 'inorder' dan 'preorder traversals' T menghasilkan berikut:]

Inorder traversal (I) : E A C K F H D B G
Preorder traverse (P) : F A E K C D H G B

[4 Marks/Markah]

c) Extract and Compute a minimum cost spanning tree (using Prims Algorithm) for the graph in *Figure 4.1* .

[Ekstrak dan Kirakan kos minimum pohon merentang (menggunakan Algoritma Prims) untuk graf dalam Rajah 4.1.]

[6 Marks/Markah]

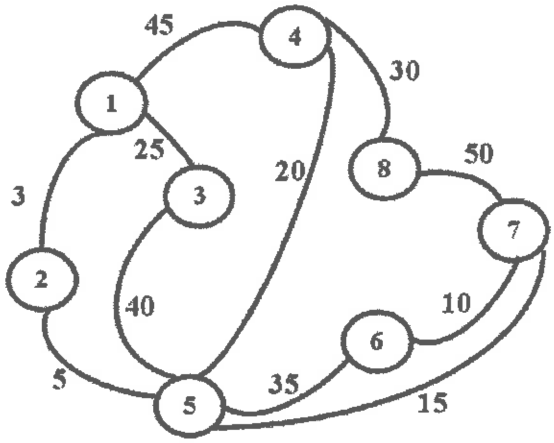

Figure 4.1
[Rajah 4.1]

Answer 4(c) Table Hints:

Edge	Cost of the Edge	V'	E'	Minimum Cost Spanning Tree

d) Consider an undirected graph G shown in *Figure 4.2*. Demonstrate the Breadth First Traversal BFT(1) on the graph G, where the start vertex is 1.

[Pertimbangkan suatu G graf tak berarah yang ditunjukkan dalam Rajah 4.2. Menunjukkan Breadth First Traversal BFT (1) pada graf G, dimana puncak permulaan ialah 1.]

[6 Marks/Markah]

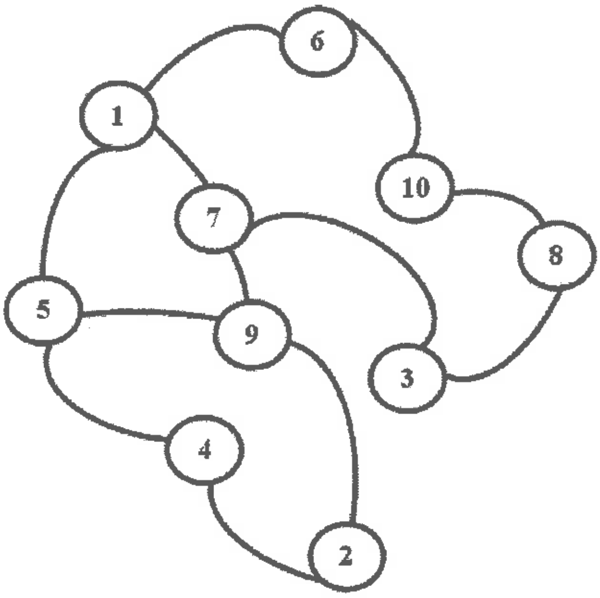

Figure 4.2

[Rajah 4.2]

Answer 4(d) Table Hints:

Current Vertex	Queue Q [1:6]						Traversal Output	Status of visited flag of vertices {1,2,3,4,5,6,7,8,9,10} of graph G									
								1	2	3	4	5	6	7	8	9	10

Dicetak oleh Unit Peperiksaan & Pengijazahan, Bahagian Pengurusan Akademik, Jabatan Pendaftar.

SECTION B

[BAHAGIAN B]

QUESTION 5

[SOALAN 5]

a) Explain the advantages of binary search over sequential search.

[Terangkan kelebihan carian binari berbanding cari berjujukan.]

[4 Marks/Markah]

b) Consider for the undirected graph G (*Figure 4.2*). Demonstrate Breadth First Search (BFS) for the key 9.

[Pertimbangkan untuk graf tak berarah G (Rajah 4.2). Tunjukkan Breadth First Search (BFS) untuk kunci 9.]

[6 Marks/Markah]

Answer 5(b) Table Hints:

Search Key	Current vertex	Queue [1:6]						Status of the visited flag(0/1) of the vertices (1-10) for graph G									
9	1 (Start)																

c) For the ordered list L = {B, D, F, G, H, I, K, L, M, N, O, P, Q, T, U, V, W, X,Y, Z}. Compare Interpolation search and Binary search for the key H and Y.

[Bagi senarai L disusun = {B, D, F, G, H, I, K, L, M, N, O, P, Q, T, U, V, W, X, Y, Z}. Bandingkan carian Interpolasi dan carian binari untuk kunci H dan Y.]

[10 Marks/Markah]

QUESTION 6

[SOALAN 6]

a) Distinguish between bubble sort and quick sort.

[Bezakan antara isihan gelembung dan isihan cepat.]

[4 Marks/Markah]

b) Demonstrate 3-way Merging on the lists:

[Tunjukkan perhimpunan 3-hala pada senarai:]

L1 = {123, 678, 345, 225, 890, 345, 111}
L2 = {345, 123, 654, 789, 912, 144, 267, 909, 111, 324}
L3 = {567, 222, 111, 900, 545, 897}

[6 Marks/Markah]

c) Consider an unsorted list L = {92, 78, 34, 23, 56, 90, 17, 52, 67, 81, 18, 92}.Compare Insertion sort and Selection sort on the list L.

[Pertimbangkan senarai L tidak terisih = {92, 78, 34, 23, 56, 90, 17, 52, 67, 81, 18, 92}. Bandingkan Isihan Kemasukkan dan Isihan Pemilihan pada senarai L.]

[10 Marks/Markah]

--- oooooOooooo---