UNIVERSITI SAINS MALAYSIA

Peperiksaan Kursus Semasa Cuti Panjang
Sidang Akademik 1997/98

April 1998

CAP102/CMP102 - Pengaturcaraan Lanjutan dan Struktur Data

CSC122 - Penyelesaian Masalah dan Pengaturcaraan

Masa : [3 jam]

ARAHAN KEPADA CALON:

- Sila pastikan bahawa kertas peperiksaan ini mengandungi **EMPAT** soalan di dalam **LAPAN** muka surat yang bercetak sebelum anda memulakan peperiksaan ini.

- Jawab **SEMUA** soalan dalam Bahasa Malaysia.

---

1. (a) (i) Berikan lima kaedah yang boleh digunakan untuk menjejak (menyurih) atur cara (program tracing).

(20/100)

(ii) Jelaskan melalui contoh yang sesuai perbezaan-perbezaan antara "stub" dan pemandu.

(20/100)

(b) (i) Tentukan apakah yang dilakukan oleh fungsi rekursi berikut:

```
int  func (int n)
{
    if   (n == 0)
        return (0);
    return (n + func (n-1));
}
```

(10/100)

(ii) Tulis semula fungsi di atas dalam bentuk lelaran (iterative).

(10/100)

(iii) Surih (simulasi) perlaksanaan func(4) dengan menunjukkan kandungan tindanan pada setiap langkah surihan.

(10/100)

(iv) Bagi fungsi 1(b)(i), kaedah manakah yang lebih sesuai untuk menulis fungsi tersebut, melalui rekursi atau lelaran (iterative)? Jelaskan jawapan anda.

(10/100)

(c) (i) Satu tindanan diwakilkan oleh satu tatasusunan S[MAXSAIZ]. S[0] menyimpan indeks data paling atas tindanan. S[1] hingga S[MAXSAIZ - 1] mengandungi unsur-unsur dalam tindanan.

Berikan prototaip dan definisi untuk fungsi-fungsi asas berikut:

- Pop (10/100)
- Push (10/100)

* Diberikan **fungsi piawai** untuk Pop dan Push sebagai panduan untuk anda menjawab soalan ini.

```
void Pop (StackEntry *item, Stack *s)
{
    if   (StackEmpty (s))
        Error ("Stack is Empty");
    else
        *item = s ∅ entry [--s ∅ top];
}
```

```
void Push (StackEntry item, Stack *s)
{
    if   (StackFull (s))
         Error ("Stack is full");
    else
         s ∅ entry [s ∅ top ++] = item;
}
```

2. (a) Jika Q ialah satu giliran, S ialah satu tindanan dan diberi prototaip untuk fungsi-fungsi berikut:

Boolean Empty (**Stack** *);
Boolean Full (**Stack** *);
void Initialize (**Stack** *);
void Push (**StackEntry**, **Stack** *);
void Pop (**StackEntry** *, **Stack** *);

void AddQueue (**StackEntry**, **Queue** *);
void DeleteQueue (**StackEntry** *, **Queue** *);
void Initialize (**Queue** *);
int Size (**Queue** *);
Boolean Empty (**Queue** *);
Boolean Full (**Queue** *);

(Anda boleh menambah pemboleh ubah baru pada fungsi jika diperlukan.)

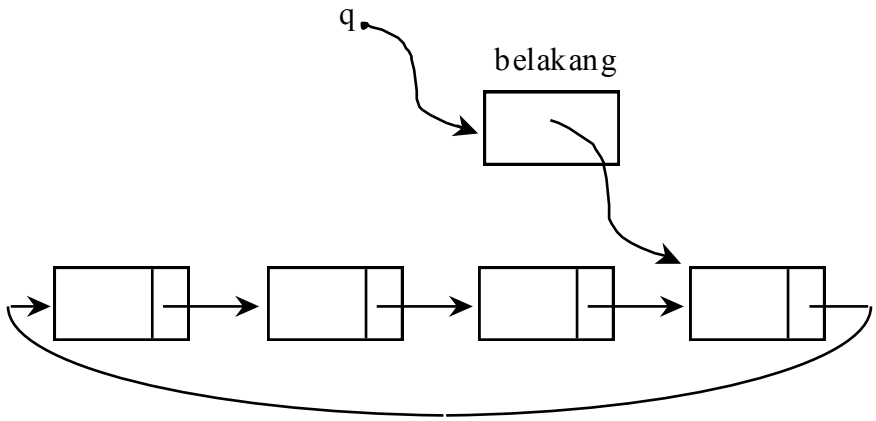
Gunakan operasi-operasi asas tindanan yang diberi untuk menulis algoritma untuk melaksanakan tugas-tugas berikut:

(i) Hapuskan semua kewujudan item X dalam giliran, tinggalkan semua unsur giliran yang lain dalam susunan asal.

(ii) Bermula dengan tindanan yang kosong dan satu giliran yang tidak kosong, terbalikkan semua kemasukan dalam giliran dengan menggunakan tindanan.

(30/100)

(b) Satu senarai berpaut membulat, seperti yang ditunjukkan dalam gambar rajah di bawah, adalah satu senarai berpaut dengan nod di belakang (rear) akan menuding kembali ke depan senarai (front). Kita hanya memerlukan satu penuding untuk mencapai depan dan belakang senarai kerana belakang ∅ next adalah depan senarai.



Jika giliran diimplementasikan dengan menggunakan senarai berpaut membulat di atas, kita hanya perlu satu penuding untuk mencapai depan dan belakang senarai.

(i) Beri struktur data giliran melalui perwakilan senarai berpaut membulat di atas.

(10/100)

(ii) Berikut adalah **fungsi piawai** Append:

```
void Append (Queuenode *p, Queue *q)
{
    if   (!p)
         printf ("Nod Tidak Wujud");
    else if (QueueEmpty (q))
              q ∅ belakang = q ∅ depan = p;
         else {
              q ∅ belakang ∅ next = p;
              q ∅ belakang = p;
              }
}
```

Tulis kembali fungsi Append ini dengan mengambil kira perwakilan dalam (i) di atas.

(20/100)

(c) Diberi fungsi SelitNod iaitu satu fungsi yang menerima data **x**, senarai **list** dan menyisipkan data **x** ke dalam senarai dan hasilnya senarai terisih mengikut susunan menurun. Lengkapkan fungsi ini.

```
void SelitNod (ListEntry x, List *list)
{
    ListNode *newnode, *current, *previous;
    newnode = malloc (sizeof (ListNode));

if  (! newnode) {
    Error ("Tiada ingatan");
    return (-1);
    }
else
    {

        newnode ∅ entry = x;
        newnode ∅ next = NULL;
        if  (list ∅ head == NULL
            {
                    newnode ∅ next = list ∅ head;
                    list ∅ head = newnode;
            }
else
    {

        (i)   /*  cari kedudukan untuk penyisipan */
              _____________________________

              _____________________________

              _____________________________


        (ii)  /*  menyisip data baru */

              _____________________________

              _____________________________

              _____________________________

    }
        list ∅ count ++;
}
```

(20/100)

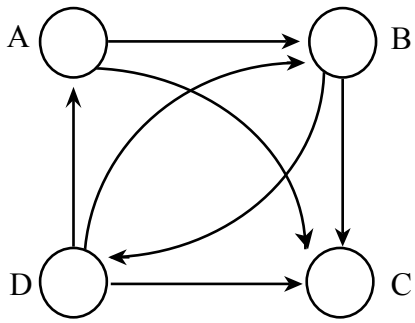
(d) Beri perbezaan antara operasi AddQueue bagi implementasi berdampingan (menggunakan tatasusunan) dan implementasi senarai berpaut.

(20/100)

3. (a) Surih Algoritma Gelintaran Berjujukan, dengan menggunakan data berikut: kunci yang dicari ialah 34. Senarai A [ ] = { 12, 16, 21, 31, 34, 42, 49, 51, 76, 78, 93 }.

(20/100)

(b) Diberi graf berarah berikut:



Ilustrasi perwakilan graf di atas dengan menggunakan perlaksanaan berikut:

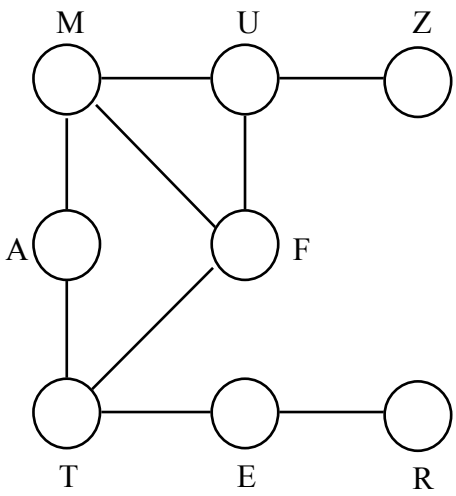
(i) jadual kesebelahan

(ii) senarai berpaut

(iii) senarai berdampingan

(30/100)

(c) Diberi graf berikut:



Susur graf di atas dengan menggunakan kaedah:

(i) Penyusuran kedalaman dahulu (DFS)

(ii) Penyusuran kelebaran dahulu (BFS)

Ilustrasikan langkah demi langkah.

(20/100)

(d) Tulis algoritma bagi kedua-dua kaedah penyusuran dalam (c) di atas.

(30/100)

4. (a) (i) Berikan algoritma bagi isihan pilih dengan menggunakan perwakilan senarai berdampingan.

(10/100)

(ii) Surih algoritma isihan pilih yang telah diberi dalam 4(a)(i) di atas dengan menggunakan data-data berikut:

26 33 35 29 19 12 22

(10/100)

(iii) Berapakah bilangan perbandingan yang diperlukan bagi mengisih set data di atas?

(5/100)

(b) Jika Isihan Cepat ingin diubahsuaikan untuk mengisih data secara menurun (bukan secara menaik), ubahsuaikan algoritma piawai berikut:

```
void QuickSort (List *list)
{
    RecQuickSort(list, 0, list ∅ count-1);
}

void RecQuickSort (List *list, Position Low, Position High)
{
    Position pivotpos;
    if (low < high) {
        pivotpos = Partition (list, low, high);
        RecQuickSort (list, low, pivotpos -1);
        RecQuickSort (list, pivotpos +1, high);
    }
{

Position Partition (List *list, Position Low, Position High)
{
    ListEntry pivot;
    Position i, lastsmall, pivotpos;
    Swap(low, (low+high)/2, list);
    pivot = list->entry [low];
    pivotpost = low;
    for (i = low + 1; i <= high; i++)
        if (LT(list -> entry [i].key, pivot.key))
            Swap(++ pivotpos, i, list);
    Swap(low, pivotpos, list);
  return pivotpos;
}
```

Contoh output bagi algoritma yang diubahsuaikan:

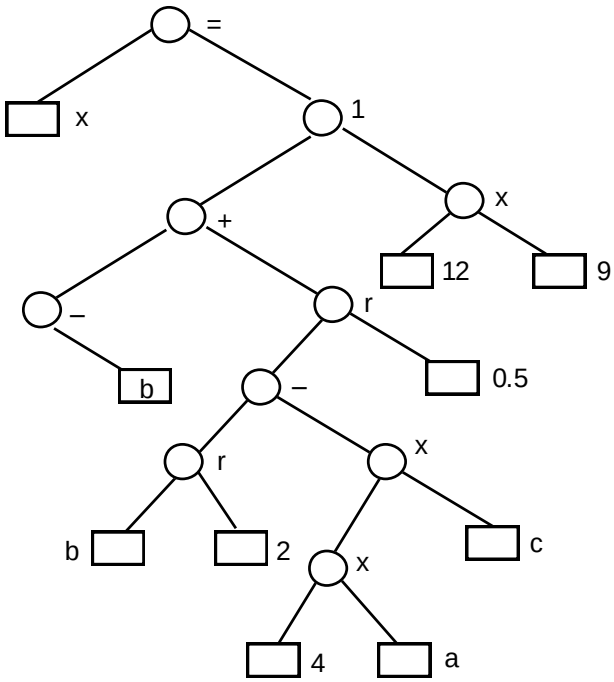
Senarai asal	∅	5	7	4	9	11	8
Senarai akhir	∅	11	9	8	7	5	4

(25 markah)

(c) Tulis algoritma bagi penyusuran tertib sisipan, tertib awalan dan tertib akhiran ke atas pepohon perduaan dengan menggunakan perwakilan senarai berpaut.

(20 markah)

(d) Jika diberikan pepohon ungkapan (expression tree) untuk rumus kuadratik berikut:



Berikan hasil penyusuran bagi kaedah berikut:

(i) Tertib sisipan

(ii) Tertib awalan

(iii) Tertib akhiran

(30 markah)

- oooOooo -