Đại học Quốc Gia TP. Hồ Chí Minh
Trường đại học Bách Khoa
Khoa: **Khoa học & Kỹ thuật Máy tính**
Bộ môn: **Khoa học Máy tính**

ĐỀ KIỂM TRA GIỮA HỌC KỲ 1
Năm học: 2010 – 2011
Môn: **Cấu trúc dữ liệu & Giải thuật**
MSMH: 503001
Ngày thi: 31/10/2010 - Thời gian: 60 phút
(*Được sử dụng tài liệu*)

Lưu ý: Đề kiểm tra gồm 4 câu với thang điểm 11/10. Sinh viên làm đúng trên 10 điểm sẽ được làm tròn thành 10.

Câu 1: (2.5 điểm)

a. (1.5 điểm) Hãy cho biết độ phức tạp của các hàm sau (theo Big-O Notation) trong trường hợp xấu nhất (chỉ ghi kết quả, không cần giải thích)

```
void ExA(int n) {
   int a;
   for (int i = 0; i < n; i += 2)
      a = i;
}

void ExB(int n) {
   int a;
   for (int i = 0; i < n * n; i++)
      a = i;
}

void ExC(int n) {
   int a;
   for (int i = 0; i < n; i++)
      for (int j = 0; j <= i; j++)
         a = i;
}

void ExD(int n) {
   int a;
   for (int i = 0; i < n * n; i++)
      for (int j = 0; j <= i; j++)
         a = i;
}

void ExE(int n) {
   int a;
   for (int i = 0; i < n; i++)
      for (int j = 0; j <= n - i; j++)
         a = i;
}

void ExF(int n) {
   int a;
   for (int i = 1; i < n; i *= 2)
      a = i;
}
```

b. (1 điểm) Cho ví dụ về hai hàm f_1 và f_2, trong đó f_1 sẽ thực thi nhanh hơn f_2 trong trường hợp tốt nhất và f_1 sẽ thực thi chậm hơn f_2 trong trường hợp xấu nhất.

Giải:
a. ExA: O(n)
ExB: $O(n^2)$
ExC: $O(n^2)$
ExD: $O(n^4)$
ExE: $O(n^2)$
ExF: $O(\log_2 n)$

b.
```
void f1(int n, int m)
{
   int a;
   if(n < m) a = n;
   else
      for(int i = 0; i < n * n; i++) a = i;
}

void f2(int n, int m)
{
   int a;
   for(int i = 0; i < n; i++) a = i;
}
```

Câu 2: (4 điểm)

Cho một cấu trúc danh sách liên kết được mô tả trong Hình 1.

```
//just an entry in the list, a "struct++" in fact
class Node {
public:
   int data;
   Node* next;
};

//interface part
class List {
private:
   int count;
   Node* pHead;
public:
   List();
   void add(int data, int index);
   void display();
   ~List();
};
```

Hình 1

Method *add* sẽ thêm *data* vào vị trí thứ *index* trong danh sách liên kết. Giả sử phần tử bắt đầu của danh sách có chỉ số là 1 và số phần tử của danh sách luôn lớn hơn *index* khi *add* được gọi.

Ví dụ : Giả sử danh sách *list* đang là (4,5,7,9). Sau khi gọi *list*.*add*(6,2) thì *list* sẽ trở thành (4,6,5,7,9).

Hãy hiện thực method *add* theo hai cách: (i) không đệ quy và (ii) đệ quy.

Giải:

a. Không đệ quy

```
void List::add(int data, int index)
{
   if(index > 0 && index < count)
   {
      Node *pCur, *pNew;
      pNew = new Node();
      pNew -> data = data;
      if(index == 1)
      {
         pNew -> next = pHead;
         pHead = pNew;
      }
      else
      {
         pCur = pHead;
         for(int i = 2; i < index; i++) pCur = pCur -> next;
         pNew -> next = pCur -> next;
         pCur -> next = pNew;
      }
      count++;
   }
}
```

b. Đệ quy

```
void List::add(int data, int index)
{
   if(index > 0 && index < count)
   {
      if(index == 1)
      {
         Node *pNew = new Node();
         pNew -> data = data;
         pNew -> next = pHead;
         pHead = pNew;
      }
      else
         addRec(data, index, 2, pHead);
      count++;
   }
}
```

```
void List::addRec(int data, int index, int position, Node* pCur)
{
   if(position == index)
   {
      Node *pNew = new Node();
      pNew -> data = data;
      pNew -> next = pCur -> next;
      pCur -> next = pNew;
   }
   else
      addRec(data, index, position + 1, pCur -> next);
}
```

Câu 3: (3 điểm)

Giả sử chúng ta đã có một cấu trúc dữ liệu *stack* đã được hiện thực cùng với các hàm sau:

boolean isEmpty(*stack s*) // kiểm tra xem *s* có rỗng hay không
int peek(*stack s*) // trả về giá trị trên đỉnh của *s*
void push(*int x*, *stack s*) // đẩy *x* vào *s*
int pop(*stack s*) // lấy phần tử đầu tiên ra khỏi *s* và trả về giá trị của phần tử này

Hãy hiện thực hàm sau: *sort*(*stack s1*, *stack s2*)
Trong đó *s1* sẽ được dùng như dữ liệu nhập, *s2* dùng như dữ liệu xuất. Sau khi *sort* thực thi, *s2* sẽ chứa các phần tử của *s1* nhưng được sắp xếp từ nhỏ đến lớn (khi đó thao tác *peek(s2)* sẽ trả về phần tử lớn nhất trong *s2*). Sinh viên lớp thường có thể khai báo các biến tạm tuỳ ý khi hiện thực hàm *sort*, sinh viên lớp KSTN **chỉ được phép khai báo thêm các biến tạm thuộc kiểu stack**.

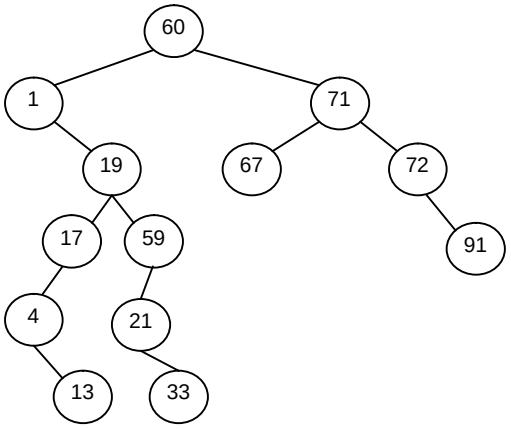
Giải:
```
void sort(stack s1, stack s2)
{
   stack stmp;
   while(!isEmpty(s1))
   {
      push(pop(s1), s2);
      while(!isEmpty(s1))
      {
         if(peek(s1) < peek(s2))
         {
            push(pop(s2), stmp);
            push(pop(s1), s2);
         }
         else push(pop(s1), stmp);
      }
      while(!isEmpty(stmp)) push(pop(stmp), s1);
   }
}
```

Câu 4: (1.5 điểm)

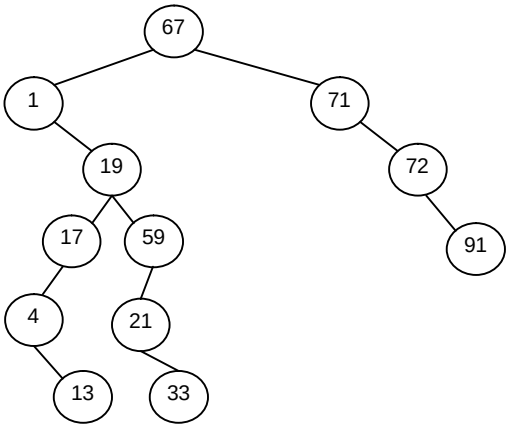
a. (1 điểm) Cho một danh sách các số nguyên như sau:
{60, 71, 1, 19, 59, 17, 4, 13, 72, 91, 67, 21, 33}
Giả sử các số nguyên này được chèn lần lượt vào một *cây nhị phân tìm kiếm* (Binary Search Tree – BST) rỗng theo đúng thứ tự trong danh sách. Hãy vẽ cây nhị phân tìm kiếm sau khi các số nguyên trong danh sách được chèn xong.

b. (0.5 điểm) Vẽ lại cây nhị phân tìm kiếm sau khi xóa node 60 từ cây nhị phân ở câu a.

Giải:
a.



b.



Sinh viên có thể dùng node 59 để thay thế.

-- Hết --