MỘT SỐ CÂU HỎI ÔN TẬP CUỐI KỲ

Câu 1:

a. Hãy trình bày ý tưởng của giải thuật tìm kiếm tuyến tính và cho biết độ phức tạp của giải thuật. *(1 điểm)*

b. Trình bày các bước (vẽ từng bước) giải thuật tìm kiếm tuyến tính thực hiện tìm giá trị X = 5 trong mảng 6 số nguyên có giá trị: 81, 90, 62, 65, 12, 42. *(1.5 điểm)*

Đáp án tham khảo:

a. – Ý tưởng: Đầu vào là một mảng/danh sách a gồm n phần tử và phần tử x là khóa cần tìm. Ta khởi gán giá trị i = 0, một vòng lặp so sánh a[i] với giá trị x cần tìm. Nếu a[i] có giá trị bằng x, thì thông báo đã tìm thấy x trong mảng/danh sách và kết thúc giải thuật. Ngược lại, ta tăng i lên 1 và xét tiếp các phần tử kế tiếp trong mảng/danh sách. Khi xét đến phần tử cuối mảng/danh sách (i = n) mà chưa tìm thấy giá trị bằng x, thì thông báo không tìm thấy x trong mảng/danh sách và kết thúc giải thuật.

– Độ phức tạp: O(n)

b. Mảng a gồm 6 phần tử như sau:

81	90	62	65	12	42

- Bước 1: Khởi tạo giá trị $i = 0 < n - 1 = 5$. So sánh $a[i] = a[0] = 81 \neq X = 5$, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 2: $i = i + 1 = 1 < 5$, so sánh $a[i] = a[1] = 90 \neq X = 5$, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 3: $i = i + 1 = 2 < 5$, so sánh $a[i] = a[2] = 62 \neq X = 5$, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 4: $i = i + 1 = 3 < 5$, so sánh $a[i] = a[3] = 65 \neq X = 5$, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 5: $i = i + 1 = 4 < 5$, so sánh $a[i] = a[4] = 12 \neq X = 5$, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 6: $i = i + 1 = 5 = 5$, so sánh $a[i] = a[5] = 42 \neq X = 5$, đã xét đến phần tử cuối cùng trong mảng, thông báo không tìm thấy phần tử X = 5 trong mảng và kết thúc thuật toán.

Câu 2:

Người ta muốn lưu trữ danh sách hàng hóa tại công ty X với các thông tin chính yếu nhằm hỗ trợ nhanh trong tra cứu, với các thông tin: Tên mặt hàng (chuỗi); Giá mặt hàng (số nguyên); Số lượng còn trong kho (số nguyên). Hãy thực hiện:

a. Định nghĩa cấu trúc dữ liệu lưu danh sách các mặt hàng theo thông tin mô tả ở trên, sử dụng cấu trúc danh sách liên kết. *(1 điểm)*

b. Viết hàm nhập vào danh sách 50 mặt hàng sử dụng cấu trúc dữ liệu ở câu 2a, biết rằng khi nhập lần lượt từng mặt hàng sẽ thêm vào cuối danh sách. *(1.5 điểm)*

c. Viết hàm nhập vào 2 số nguyên dương x, y (x < y), hiển thị lên màn hình danh sách mặt hàng có số lượng trong kho lớn hơn x và nhỏ hơn y. *(1 điểm)*

Đáp án tham khảo:

a. Cấu trúc một hàng hóa trong danh sách:

```
typedef struct HangHoa {
        char ten[100];
        int gia;
        int soluong;
        struct HangHoa* Next;
} node;
```

Cấu trúc danh sách liên kết đơn:

```
typedef struct DanhSach {
        node* Head;
        node* Tail;
} list;
```

b.
```
node* CreateNode(char x, int y, int z) {
    node* p;
    p = new node;
    if (p == NULL) {
        cout << "Khong du bo nho";
        return NULL;
    }
    p→ten = x;
    p→gia = y;
    p→soluong = z;
    p→Next = NULL;
    return p;
}

void AddTail(list &l, node *p) { // thêm phần tử vào cuối danh sách
    if (l.Head == NULL)
        l.Head = p;
        l.Tail = p;
    else {
        l.Tail→Next = p;
        l.Tail = p;
    }
}
```

```
    void Nhap(list &l) {
        int gia, soluong;
        node* p;
        char ten[100];
        for (int i = 0; i < 50; i++)
        {
                cout << "Nhap ten mat hang: ";
                fgets(ten, 100, stdin);
                cout << "Nhap gia mat hang: ";
                cin >> gia;
                cout << "Nhap so luong con trong kho: ";
                cin >> soluong;
                p = CreateNode(ten, gia, soluong);
                AddTail(l, p);
        }
    }
c.  void Xuat(list &l) {
        int x, y;
        do
        {
                cout << "Nhap so nguyen duong x: ";
                cin >> x;
                if (x <= 0)
                        cout << "Gia tri x khong hop le, vui long nhap lai!";
        } while(x <= 0);
        do {
                cout << "Nhap so nguyen duong y: ";
                cin >> y;
                if (y <= 0)
                        cout << "Gia tri y khong hop le, vui long nhap lai!";
        } while(y <= 0);
        node* p;
        p = l.Head;
        while (p) {
                if (p→soluong > x && p→soluong < y) {
                        cout << p→ten << "      ";
                        p = p→Next;
                }
        }
}
```

Câu 3:
Hãy thực hiện chuyển đổi một số nguyên dương N (N < 1000) ở hệ thập phân sang biểu diễn ở hệ nhị phân (ví dụ: số 5 ở hệ thập phân sẽ là 101 ở hệ nhị phân), sử dụng cấu trúc ngăn xếp (stack), với các yêu cầu sau:

a. Định nghĩa cấu trúc ngăn xếp để lưu trữ số nhị phân. *(1 điểm)*
b. Viết các hàm thao tác với cấu trúc ngăn xếp trong câu 3a: push, pop, kiểm tra stack rỗng, kiểm tra stack đầy. *(2 điểm)*
c. Viết hàm nhận đầu vào một số nguyên dương N ở hệ thập phân, chuyển đổi và hiển thị kết quả số N ở hệ nhị phân lên màn hình sử dụng cấu trúc, các hàm đã định nghĩa trong câu 3a, 3b. *(1 điểm)*

Đáp án tham khảo:

a. *(Sinh viên có thể sử dụng mảng hoặc danh sách liên kết để cài đặt Stack)*

- ***Cách 1:*** Sử dụng mảng, có khai báo kích thước tối đa của mảng.

```
#define MAX 100
struct stack
{
    int n;
    int arr[MAX];
};
```

- ***Cách 2:*** Sử dụng danh sách liên kết

```
struct node
{
    int data;
    struct node* next;
};
typedef node* pnode;
```

b. Sử dụng mảng:

```
void Init(stack &s)
{
        s.n = -1;
}

int IsEmpty(stack s)
{
        if (s.n == -1)
                return 1;
        else return 0;
}
```

```
int IsFull(stack s)
{
	return (s.n == MAX – 1);
}

void push(stack &s, int x)
{
	if (IsFull(s))
		cout << “Ngan xep day!”
	else {
		s.n++;
		s.arr[s.n] = x;
	}
}

int pop(stack &s)
{
	if (IsEmpty(s)) {
		cout << “Ngan xep rong!”;
		return -1;
	}
	else return s.arr[s.n--];
}
```

Sử dụng danh sách liên kết: *(không tồn tại hàm kiểm tra stack đầy)*

```
void Init(pnode &s)
{
	s = NULL;
}

int IsEmpty(pnode s)
{
	return (s = NULL);
}

void push(pnode &s, int x)
{
	pnode = p;
	p = new node;
	p→data = x;
	p→next = s;
```

```
        s = p;
    }

    int pop(pnode &s)
    {
        int x;
        pnode = p;
        x = s→data;
        p = s;
        s = s→next;
        delete p;
        return x;
    }
```

c. Sử dụng mảng:

```
    void Convert_10_2(int n)
    {
        stack s;
        Init(s);
        int k;
        while (n != 0)
        {
            k = n % 2;
            push(s, k);
            n = n / 2;
        }
        cout << "So nhi phan la: ";
        while (IsEmpty(s) == 0)
            cout << pop(s);
    }
```

Sử dụng danh sách liên kết:

```
    void Convert_10_2(int n)
    {
        pnode p;
        Init(p);
        int x;
        while (n != 0) {
            x = n % 2;
            push(p, x);
            n = n / 2;
        }
```

```
        cout << "So nhi phan la: ";
        while (IsEmpty(p) == 0)
            cout << pop(p);
    }
```

Câu 4: ***(4.5 điểm)***
Cho chuỗi S (ví dụ S = "ABCD"), hãy in ra màn hình chuỗi đảo ngược của chuỗi S ("DCSA") bằng cách sử dụng cấu trúc ngăn xếp (stack). Hãy:

a. Định nghĩa cấu trúc dữ liệu biểu diễn ngăn xếp theo yêu cầu:
b. Viết hàm in ra chuỗi đảo ngược của chuỗi S và các hàm khác có liên quan, sử dụng cấu trúc dữ liệu trong câu 1a.

Đáp án tham khảo:

a.
```
struct node {
    char data;
    node *next;
};
typedef node *pnode;
```

b.
```
void Init(pnode &s) {
    s = NULL;
}

int IsEmpty(pnode s) {
    return (s == NULL);
}

void Push(pnode &s, char x) {
    pnode p;
    p = new node;
    p→data = x;
    p→next = s;
    s = p;
}

int Pop(pnode &s) {
    char x;
    pnode p;
    x = s→data;
    p = s;
    s = s→next;
```

```
    delete p;
    return x;
}

void DaoNguoc(char s[]) {
    int n = strlen(s);
    pnode top = NULL;
    for (int i = 0; i < n; i++) {
        Push(top, s[i]);
    }
    cout << "Chuoi dao nguoc: ";
    while (top != NULL) {
        cout << Pop(top) << endl;
    }
}
```

Câu 5: ***(5.5 điểm)***

Giả sử người ta sử dụng một danh sách liên kết đơn gồm N phần tử (1 < N < 100) để cài đặt ứng dụng quản lý mua vé xem phim, mỗi phần tử trong danh sách là thông tin của một người mua vé xem phim, bao gồm các thông tin: Họ tên, số lượng vé muốn mua, loại ghế (1: ghế đơn, 2: ghế đôi), ngày, tháng xem phim, loại vé (1: giảm giá, 0: không giảm giá). Hãy:

a. Định nghĩa cấu trúc dữ liệu cho danh sách liên kết.
b. Viết hàm nhập vào danh sách M người mua vé, M nhập từ bàn phím (0 < M < N)
c. Viết hàm hiển thị danh sách người mua vé loại không giảm giá.
d. Viết hàm đếm số lượng người mua vé chọn loại ghế ngồi là ghế đôi.

Đáp án tham khảo:

a.
```
typedef struct ThongTin {
    char hoten[50];
    int soluongve;
    int loaighe;
    string ngaythang;
    int loaive;
    struct ThongTin *next;
} node;
struct list {
    node *head;
    node *tail;
}
```

b.
```
node* CreateNode(char a, int b, int c, string d, int e) {
    node *p;
    p = new node;
    if (p == NULL) {
        cout << “Khong du bo nho!”;
        return NULL;
    }
    p→hoten = a;
    p→soluongve = b;
    p→loaighe = c;
    p→ngaythang = d;
    p→loaive = e;
    p→next = NULL;
    return p;
}
void AddHead (list &l, node *p) {
    if (l.head = NULL) {
        l.head = p;
        l.tail = p;
    }
    else {
        p→next = l.head;
        l.head = p;
    }
}
void NhapDanhSach(list &l) {
    node *p;
    char hoten[50];
    string ngaythang;
    int soluong, loaighe, loaive, m;
    do {
        cout << “Nhap so nguoi mua ve can nhap vao danh sach (> 0): ”;
        cin >> m;
    } while (m < 0);
    for (int i  = 0; i < m; i++)
    {
        cin.ignore();
        cout << “Nhap ho ten nguoi thu ” << i + 1;
        cin >> hoten;
        cout << “Nhap so luong ve: ”;
```

```
            cin >> soluong;
            cout << "Nhap thong tin loai ghe: ";
            cin >> loaighe;
            cout << "Nhap ngay, thang mua ve: ";
            getline(cin, ngaythang);
            cin.ignore();
            cout << "Nhap thong tin loai ve: ";
            cin >> loaive;
            p = CreateNode(hoten, soluong, loaighe, ngaythang, loaive);
            AddHead(l, p);
        }
    }
```

c.
```
void HienThiNguoiMuaVeLoai0(list &l) {
    node *p;
    p = l.head;
    while (p != NULL)
    {
        if (p→loaive == 0)
            cout << p→hoten << "\n";
        p = p→next;
    }
}
```

d.
```
void DemSoNguoiMuaGheDoi(list &l) {
    node *p;
    p = l.head;
    int count = 0;
    while (p != NULL)
    {
        if (p→loaighe == 2)
            count++;
        p = p→next;
    }
    cout << "So luong nguoi mua ve ghe doi la: " << count << endl;
}
```

Câu 6: Cho mảng 1 chiều số nguyên A sau gồm 7 phần tử: 3, 7, 2, 1, 2, 5, 8

a. Hãy cho biết quá trình tính toán khi tìm trong mảng A phần tử có giá trị bằng 6 của thuật toán tìm kiếm tuyến tính của mảng A.
b. Trình bày ý tưởng và code của thuật toán tìm kiếm tuyến tính (cải tiến)

Đáp án tham khảo:

a. Mảng a gồm 7 phần tử:

3	7	2	1	2	5	8

- Bước 1: Khởi tạo giá trị i = 0 < n – 1 = 6. So sánh a[i] = a[0] = 3 ≠ X = 6, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 2: i = i + 1 = 1 < 6, so sánh a[i] = a[1] = 7 ≠ X = 6, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 3: i = i + 1 = 2 < 6, so sánh a[i] = a[2] = 2 ≠ X = 6, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 4: i = i + 1 = 3 < 6, so sánh a[i] = a[3] = 1 ≠ X = 6, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 5: i = i + 1 = 4 < 6, so sánh a[i] = a[4] = 2 ≠ X = 6, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 6: i = i + 1 = 5 < 6, so sánh a[i] = a[5] = 5 ≠ X = 6, chuyển qua xét phần tử kế tiếp trong mảng.
- Bước 7: i = i + 1 = 6 = 6, so sánh a[i] = a[6] = 8 ≠ X = 6, đã xét đến phần tử cuối cùng trong mảng, thông báo không tìm thấy phần tử X = 6 và kết thúc giải thuật.

b. Đầu vào là một mảng/danh sách a gồm n phần tử và phần tử x là khóa cần tìm. Ta khởi gán giá trị i = 0, một vòng lặp so sánh giá trị a[i] với x, nếu a[i] < x ta tăng biến đếm i lên 1 và duyệt các phần tử kế tiếp trong mảng/danh sách. Nếu a[i] bằng với giá trị x cần tìm thì trả về i, thông báo đã tìm thấy vị trí của x và thoát vòng lặp. Nếu đã xét đến phần tử cuối cùng mà vẫn chưa có phần tử nào bằng giá trị với x thì thông báo không tìm được x, trả về giá trị 0 và kết thúc thuật toán.

```
int TKTTCaiTien(int *a, int n, int x) {
        int i = 0;
        a[n] = x;
        while (a[i] < x) {
                i++;
                if (i < n && a[i] == x)
                        return i;
        }
        return 0;
}
```

Câu 7: Cho hàm main như sau:

```
int main()
{
        Queue Q;
```

```
        CreateEmptyQueue(Q);
        EnQueue(Q,1);
        EnQueue(Q,2);
        DeQueue(Q);
        EnQueue(Q,3);
        EnQueue(Q,4);
        EnQueue(Q,5);
        DeQueue(Q);
        EnQueue(Q,6);
        DeQueue(Q);
        DeQueue(Q);
        PrintQueue(Q);
        return 0;
}
```

a. Hãy thực hiện các khai báo cấu trúc và định nghĩa hàm cần thiết để main thực thi.
b. Cho biết kết quả xuất ra màn hình của hàm main dựa vào kết quả của hàm PrintQueue.

Đáp án tham khảo:

a.
```
struct node {
    int data;
    node* next;
};
struct Queue {
    node* front;
    node* back;
};
void CreateEmptyQueue(Queue &q) {
    q.front = NULL;
    q.back = NULL;
}
bool IsEmpty(Queue q) {
    return (q.front == NULL);
}
void EnQueue(Queue &q, int x) {
    node* p;
    p = new node;
    p→data = x;
    p→next = NULL;
    if (q.back == NULL) {
```

```
            q.front = p;
            q.back = p;
        }
        else {
            q.back→next = p;
            q.back = p;
        }
    }
    void DeQueue(Queue &q) {
        if (IsEmpty(q)) {
            return;
        }
        node* p = q.front;
        q.front = q.front→next;
        if (q.front == NULL) {
            q.back = NULL;
        }
        delete p;
    }
    void PrintQueue(Queue q) {
        node* p = q.front;
        while (p != NULL) {
            cout << p→data << "      ";
            p = p→next;
        }
    }
```

b. Kết quả xuất ra màn hình của hàm main dựa vào kết quả của hàm PrintQueue() là: 5 6

Câu 8:

Viết chương trình quản lý BÃI ĐỖ XE với thêm thông tin trong bãi đỗ xe gồm:

- Biển số xe: kiểu chuỗi
- Loại xe: kiểu số nguyên (1: xe máy, 2: xe ô tô)
- Thời gian đỗ trong bãi: kiểu số nguyên

Ví dụ: Biển số xe: 59X2 – 000.01
Loại xe: 1 (xe máy)
Thời gian đỗ: 3 (giờ)

a. Hãy thực hiện định nghĩa cấu trúc dữ liệu cần thiết để lưu trữ các thông tin được mô tả ở trên, sử dụng **danh sách liên kết đơn.**

b. Viết hàm thống kê số lượng xe máy và xe ô tô có trong bãi xe.

c. Tính doanh số của bãi đỗ xe dựa vào số xe trong bãi hiện có. Biết phí đỗ xe được tính như sau:
   - Xe máy: thời gian đỗ <= 10 giờ là 5.000 VNĐ, sau 10 giờ thì cứ thêm 1 giờ sẽ tính thêm 1.000 VNĐ
   - Xe ô tô: thời gian đỗ <= 10 giờ là 10.000 VNĐ, sau 10 giờ thì cứ thêm 1 giờ sẽ tính thêm 1.000 VNĐ

Đáp án tham khảo:

a.
```
typedef struct BaiDoXe {
    string BienSoXe;
    int LoaiXe;
    int ThoiGianDoXe; // tính theo giờ
    struct BaiDoXe* next;
} node;
struct list {
    node* head;
    node* tail;
};
```

b.
```
void ThongKe(list &l) {
    node *p;
    p = l.head;
    int count_xemay = 0, count_oto = 0;
    while (p != NULL) {
        if (p→LoaiXe == 1)
            count_xemay++;
        else if (p→LoaiXe == 2)
            count_oto++;
        p = p→next;
    }
    cout << “So luong xe may: ” << count_xemay << endl;
    cout << “So luong xe oto: ” << count_oto << endl;
 }
```

c.
```
void TinhDoanhSo(list &l) {
    int tong1 = 0, tong2 = 0;
    node* p;
    p = l.head;
    while (p != NULL) {
        if (p→LoaiXe == 1) {
            if (p→ThoiGianDoXe <= 10) {
                tong1 += 5000;
```

```
                }
                else {
                        tong1 += 5000 + (p→ThoiGianDoXe – 10) * 1000;
                }
        else if (p→LoaiXe == 2) {
                if (p→ThoiGianDoXe > 10) {
                        tong2 += 10000;
                }
                else {
                        tong2 += 10000 + (p→ThoiGianDoXe – 10) * 1000;
                }
        p = p→next;
    }
    cout << "Tong doanh thu cua bai do xe: " << tong1 + tong2 << " VNĐ";
}
```

Câu 9: Cho mảng 1 chiều số nguyên A sau gồm 7 phần tử: 1, 2, 3, 5, 7, 8, 9

a. Hãy cho biết quá trình tính toán khi tìm trong mảng A phần tử có giá trị bằng 6 của thuật toán tìm kiếm nhị phân áp dụng cho mảng A có thứ tự giảm dần.
b. Trình bày ý tưởng và code của thuật toán tìm kiếm nội suy.

Đáp án tham khảo:

a. Mảng a[] gồm 7 phần tử:

1	2	3	5	7	8	9

- Bước 1: Khởi tạo giá trị left = 0, right = 6.
- Bước 2: Khởi tạo giá trị mid = (left + right) / 2 = (0 + 6) / 2 = 3. So sánh giá trị a[mid] = a[3] = 5 < 6 → left = mid + 1 = 4.
- Bước 3: mid = (4 + 6) / 2 = 5, a[5] = 8 > 6 → right = mid – 1 = 4.
- Bước 4: mid = (4 + 4) / 2 = 4. a[4] = 7 > 6 → right = mid – 1 = 3.
- Bước 5: left > right (3 > 4), thông báo ra màn hình không tìm thấy giá trị X = 6 và kết thúc thuật toán.

b. Đầu vào là danh sách/mảng a gồm n phần tử và X là khóa cần tìm, khởi tạo giá trị left, right lần lượt là vị trí thấp nhất và cao nhất. Khởi tạo giá trị mid = left + [(right – left) / (a[right] – a[left])] * (X – a[left]). Nếu phần tử cần tìm có giá trị lớn hơn phần tử ở giữa thì phần tử cần tìm sẽ ở mảng con bên phải phần tử ở giữa và chúng ta lại tiếp tục tính vị trí dò; nếu không phần tử cần tìm sẽ ở mảng con bên trái phần tử ở giữa. Tiến trình này tiến tụp diễn ra trên các mảng con cho tới khi kích cỡ của mảng con giảm về 0.

```
int InterPolationSearch(int arr[], int n, int x) {
        int left = 0;
```

```
        int right = n-1;
        while (left <= right && x >= arr[left] && x <= arr[right]) {
            double val1 = (double) (x - arr[left]) / (arr[right]-
arr[left]);
            int val2 = (right-left);
            int pos = left + val1*val2;
            if (arr[pos] == x)
                return pos;
            if (arr[pos] < x)
                left = pos + 1;
            else
                 right = pos - 1;
        }
        return -1;
}
```

Câu 10: Cho hàm main như sau:

```
int main()
{
    Stack S;
    CreateEmptyStack(S);
    Pop(S,1);
    Pop (S,2);
    Push(S);
    Pop (S,3);
    Pop (S,4);
    Pop (S,5);
    Push (S);
    Pop (S,6);
    Push (S);
    Push (S);
    PrintStack(S);
    return 0;
}
```

a. Hãy thực hiện các khai báo cấu trúc và định nghĩa hàm cần thiết để main thực thi.
b. Cho biết kết quả xuất ra màn hình của hàm main dựa vào kết quả của hàm PrintStack.

Đáp án tham khảo:

a. #define MAX_SIZE 10

```
typedef struct {
    int data[MAX_SIZE];
    int top;
} Stack;

void CreateEmptyStack(Stack *s) {
    s→top = -1;
}

int IsEmpty(Stack *s) {
    return (s→top == -1);
}

int IsFull(Stack *s) {
    return (s→top == MAX_SIZE – 1);
}

void Push(Stack *s, int value) {
    if (IsFull(s)) {
            cout << “Stack da day”;
            return;
    }
    s→top++;
    s→data[s→top] = value;
}

int Pop(Stack *s) {
    if (IsEmpty(s)) {
            cout << “Stack rong.”;
            return -1;
    }
    int value = s→data[s→top];
    s→top--;
    return value;
}
```

b. Kết quả xuất ra màn hình của hàm main dựa vào kết quả của hàm PrintStack():
2 5 4 6

Câu 11:
Viết chương trình quản lý PHÒNG GAME với thông tin trong phòng game gồm:

- ID máy: kiểu chuỗi
- Loại máy: kiểu số nguyên (1: Laptop, 2: PC)
- Thời gian chơi: kiểu số nguyên (đơn vị tính phút)

Ví dụ: ID máy: L001
Loại máy: 1
Thời gian chơi: 300 (phút)

a. Hãy thực hiện định nghĩa cấu trúc dữ liệu cần thiết để lưu trữ các thông tin được mô tả ở trên, sử dụng **danh sách liên kết đơn/kép.**
b. Viết hàm thống kê số lượng LAPTOP và PC có trong phòng game.
c. Tính doanh số của phòng game dựa vào thời gian chơi của các máy. Biết phí sử dụng máy trong phòng game được tính như sau:
   - Laptop: 1 giờ = 12.000 VNĐ
   - PC: 1 giờ = 6.000 VNĐ

a.
```
typedef struct PhongGame {
    string IDmay;
    int LoaiMay;
    int ThoiGianChoi; // tính theo phút
    struct BaiDoXe* next;
} node;
struct list {
    node* head;
    node* tail;
};
```

b.
```
void ThongKe(list &l) {
    node *p;
    p = l.head;
    int count_laptop = 0, count_pc = 0;
    while (p != NULL) {
        if (p→LoaiXe == 1)
            count_laptop++;
        else if (p→LoaiXe == 2)
            count_pc++;
        p = p→next;
    }
    cout << “So luong laptop: ” << count_laptop << endl;
    cout << “So luong pc: ” << count_pc << endl;
}
```

c.
```
void TinhDoanhSo(list &l) {
    int tong1 = 0, tong2 = 0;
```

```
    node* p;
    p = l.head;
    while (p != NULL) {
        if (p→LoaiXe == 1) {
            tong1 = p→ThoiGianChoi * 12000;
        }
        else if (p→LoaiXe == 2) {
            if (p→ThoiGianDoXe > 10) {
                tong2 = p→ThoiGianChoi * 6000;
        }
        p = p→next;
    }
    cout << “Tong doanh thu cua phong game: ” << tong1 + tong2 << “ VNĐ”;
}
```