BÀI TẬP LÀM THÊM

(Lập trình bằng ngôn ngữ C hoặc C++)

A. Danh sách.

1- Xây dựng các thao tác sau với danh sách cài đặt bằng mảng có cấu trúc:

```
struct CanBo {      long MaSo; char HoTen[8];      };
struct DanhSach {   CanBo DSCB[100]; int TongSoCB; };
```

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng. đầy
3. Thêm một phần tử vào đầu danh sách.
4. Thêm một phần tử vào cuối danh sách.
5. Thêm một phần tử vào sau vị trí thứ k.
6. Thêm một phần tử vào trước vị trí thứ k
7. Xoá phần tử đầu danh sách.
8. Xoá phần tử cuối danh sách.
9. Xoá phần tử thứ k.
10. Xoá toàn bộ danh sách.
11. Xem danh sách trên màn hình
12. Tìm một phần tử theo MaSo cán bộ

2- Xây dựng các thao tác sau với danh sách liên kết đơn có cấu trúc:

```
struct CanBo {      long MaSo; char HoTen[8];      };
struct Node { CanBo Info; Node *Next; };
struct DanhSach {   Node *PFirst, *PLast; };
```

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng
3. Thêm một phần tử vào đầu danh sách.
4. Thêm một phần tử vào cuối danh sách.
5. Thêm một phần tử vào sau vị trí thứ k.
6. Thêm một phần tử vào trước vị trí thứ k

7. Xoá phần tử đầu danh sách.
8. Xoá phần tử cuối danh sách.
9. Xoá phần tử thứ k.
10. Xoá toàn bộ danh sách.
11. Xem danh sách trên màn hình
12. Tìm một phần tử theo MaSo cán bộ

3- Xây dựng các thao tác sau với danh sách liên kết kép có cấu trúc:

```
struct CanBo {      long MaSo; char HoTen[8];      };
struct Node { CanBo Info; Node *Next, *Pre; };
struct DanhSach {   Node *PFirst, *PLast; };
```

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng
3. Thêm một phần tử vào đầu danh sách.
4. Thêm một phần tử vào cuối danh sách.
5. Thêm một phần tử vào sau vị trí thứ k.
6. Thêm một phần tử vào trước vị trí thứ k
7. Xoá phần tử đầu danh sách.
8. Xoá phần tử cuối danh sách.
9. Xoá phần tử thứ k.
10. Xoá toàn bộ danh sách.
11. Xem danh sách trên màn hình
12. Tìm một phần tử theo MaSo cán bộ

4- Xây dựng các thao tác sau với ngăn xếp bằng danh sách liên kết đơn có cấu trúc:

```
struct CanBo {      long MaSo; char HoTen[8];      };
struct Node { CanBo Info; Node *Next; };
struct NganXep {    Node *PHead; };
```

1. Khởi tạo ngăn xếp
2. Kiểm tra ngăn xếp rỗng

3. Thêm một phần tử vào ngăn xếp.
4. Lấy ra một phần tử từ ngăn xếp .

5- Xây dựng các thao tác sau với hàng đợi bằng danh sách liên kết đơn có cấu trúc:

```
struct CanBo {      long MaSo; char HoTen[8];      };
struct Node { CanBo Info; Node *Next; };
struct HangDoi {    Node *PFirst, *PLast; };
```

1. Khởi tạo hàng đợi
2. Kiểm tra hàng đợi rỗng
3. Thêm một phần tử vào hàng đợi.
4. Lấy ra một phần tử từ hàng đợi .

B. Sắp xếp.

Sắp xếp dãy số thực a[] tăng dần (hoặc giảm dần) bằng các giải thuật:

1. Sắp xếp chọn: void SelectionSort(float a[], int n);
2. Sắp xếp chèn: void InsertionSort(float a[], int n);
3. Sắp xếp nổi bọt: void BubbleSort(float a[], int n);
4. Sắp xếp nhanh: void QuickSort(float a[], int n);
5. Sắp xếp trộn: void MergeSort(float a[], int n);
6. Sắp xếp vun đống void HeapSort(float a[], int n);

C. Giải thuật sinh và quay lui

(Dữ liệu có thể vào ra bàn phím màn hình hoặc vào ra trên tệp)

1. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:

(i) K là số có 5 chữ số;

(ii) K là số nguyên tố;

(iii) K là số thuận nghịch (k là số thuận nghịch nếu đọc xuôi hay đọc ngược các chữ số của k ta đều nhận được một số như nhau. Ví dụ số: 30303);

(iv) Biểu diễn của K ở hệ cơ số B (B bất kỳ được nhập từ bàn phím cũng là một số thuận nghịch. Ví dụ số k=30303 có biểu diễn ở hệ cơ số 8 là 73137 cũng là một số thuận nghịch;

2. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:

(i) K là số có 5 chữ số;

(ii) K là số nguyên tố;

(iii) Đảo ngược các chữ số trong của K cũng là một số nguyên tố;

(iv) Tổng các chữ số của K cũng là một số nguyên tố;

(v) Mỗi chữ số trong K cũng là những số nguyên tố.

3. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:

(i) K là số có 5 chữ số;

(ii) K là số nguyên tố;

(iii) Mỗi chữ số của K cũng là những số nguyên tố;

(iv) Tổng các chữ số của K là một số thuận nghịch hai chữ số;

(v) Tích các chữ số của K là một số thuận nghịch ba chữ số.

4. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:

(i) K là số có 5 chữ số;

(ii) K là số nguyên tố;

(iii) Mỗi chữ số của K cũng là các số nguyên tố;

(iv) Tổng các chữ số của K là một số chia hết cho P (P được nhập từ bàn phím);

(v) Tích các chữ số của K là một số chia hết cho Q (Q được nhập từ bàn phím);

(vi) Các chữ số của K không chứa số R (được nhập từ bàn phím).

5. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:

(i) K là số có 5, 7 hoặc 9 chữ số;

(ii) K là số thuận nghịch;

(iii) Tổng các chữ số của K là một số chia hết cho P (P được nhập từ bàn phím);

(iv) Tích các chữ số của K là một số chia hết cho Q (Q được nhập từ bàn phím);

(v) Các chữ số của K không chứa số R (được nhập từ bàn phím).

6. Cho số tự nhiên N, B được nhập từ bàn phím (N≥10000 ; B≥255). Hãy viết chương trình thực hiện:

(i) Tính tổng các chữ số của N ;

(ii) Phân tích N thành tích các thừa số nguyên tố ;

(iii) Biểu diễn N ở hệ cơ số B ;

(iv) Liệt kê các số hoàn hảo nhỏ hơn N ;

7. Số điện thoại di động của một hãng viễn thông được đánh số theo qui cách 091N. XXX.XXX. Trong đó, N là số từ 2 đến 8, X là một số từ 0 đến 9. Ta định nghĩa các loại số điện thoại sau:

- **Số điện thoại loại I (Loại I):** Là những số có sáu số cuối cùng của nó tạo thành một số thuận nghịch sáu chữ số. Ví dụ số: 0913.558855.
- **Số điện thoại loại II (Loại II):** Là những số điện thoại Loại I có tổng sáu chữ số cuối cùng là một số chia hết cho 10 . Ví dụ số: 0913.104.401 (1+0+4+4+0+1=10).
- **Số điện thoại loại III (Loại III):** Là những số điện thoại Loại II có sáu chữ số cuối cùng không chứa bất kỳ một số 0 nào. Ví dụ số: 0913. 122.2211.

Hãy viết chương trình thực hiện:

- Liệt kê tất cả các số điện thoại Loại I không chứa các số điện thoại Loại II. Ghi lại các số Loại I vào file Loai1.out theo từng dòng, mỗi dòng không quá 8 số điện thoại.
- Liệt kê tất cả các số điện thoại Loại II không chứa các số điện thoại Loại III. Ghi lại các số Loại II vào file Loai2.out theo từng dòng, mỗi dòng không quá 8 số điện thoại.
- Liệt kê tất cả các số điện thoại Loại III. Ghi lại các số Loại III vào file Loai3.out theo từng dòng, mỗi dòng không quá 8 số điện thoại.

8. Một xâu nhị phân độ dài *n* được gọi là thuận nghịch hay đối xứng nếu đảo ngược xâu nhị phân đó ta vẫn nhận được chính nó. Cho số tự nhiên *n* (n nhập từ bàn phím). Hãy viết chương trình liệt kê tất cả các xâu nhị phân thuận nghịch có độ dài *n*. Các xâu nhị phân tìm được ghi lại trong file ketqua.out theo khôn dạng:

- Dòng đầu tiên ghi lại số K là số các xâu thuận nghịch có độ dài n tìm được;
- K dòng kế tiếp ghi lại mỗi dòng một xâu nhị phân thuận nghịch có độ dài n. Hai phần tử khác nhau của xâu thuận nghịch được ghi cách nhau một vài khoảng trống.

Ví dụ với n = 4 ta tìm được 4 xâu nhị phân thuận nghịch như dưới đây.

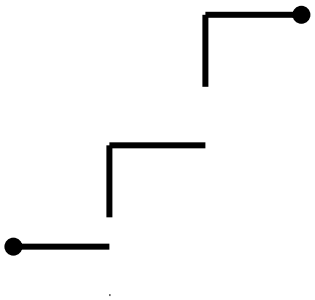
ketqua.out

```
4
0 0 0 0
0 1 1 0
1 0 0 1
1 1 1 1
```

9. Cho một hình chữ nhật gồm n × m hình vuông đơn vị (n, m được nhập từ bàn phím). Hãy liệt kê tất cả các đường đi từ đỉnh cuối của ô vuông cuối cùng phía bên trái đến đỉnh đầu của ô vuông trên cùng phía bên phải. Biết mỗi bước đi chỉ được phép dịch chuyển sang bên phải (ký hiệu là bước 1) hoặc lên trên (ký hiệu là bước 0) theo các cạnh của hình vuông đơn vị. Các đường đi tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại một số tự nhiên là số các đường đi tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một đường đi, bước dịch phải (1) và bước lên trên (0) của mỗi đường đi, hai bước khác nhau được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với n = 3, m = 2 như hình vuông dưới đây sẽ cho ta file ketqua.out tương ứng.



Ketqua.Out

```
10
0       0       1
        1       1
0       1       0
        1       1
0       1       1
        0       1
0       1       1
        1       0
1       0       0
        1       1
1       0       1
```

10. Cho dãy gồm *n* số tự nhiên phân biệt $a_1, a_2, .., a_n$ và số tự nhiên *B*. Hãy viết chương trình liệt kê tất cả các phần tử của tập $D = \left\{ (x_1, x_2, \cdots, x_n) : \sum_{i=1}^{n} a_i x_i = B, \quad x_i \in \{0,1\}, i = 1,2,..., n \right\}$;

Dữ liệu vào cho bởi file data.in theo khuôn dạng như sau:

- Dòng đầu tiên ghi lại hai số tự nhiên *n* và *B*. Hai số được viết cách nhau bởi một vài khoảng trống.
- Dòng kế tiếp ghi lại n số nguyên dương $a_1, a_2,..,a_n$. Hai số khác nhau được viết cách nhau bởi một vài kí tự trống.

Kết quả ra ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên *k* là số phần tử của tập *D*.
- k dòng tiếp theo mỗi dòng ghi lại một vector nhị phân $x = (x_1, x_2 ,.., x_n)$ là phần tử của *D*. Hai thành phần khác nhau của vector *x* được viết cách nhau bởi một vài khoảng trống.

Ví dụ với $n = 7, B = 25, \{ a_1, a_2, a_3, a_4, a_5, a_6, a_7\} = \{5, 10, 15, 20, 25, 30, 35\}$ trong file data.in sẽ cho ta *3* phần tử của tập *D* tương ứng với *3* vector nhị phân độ dài *n* trong file ketqua.out dưới đây:

Data.in

```
7       25
5       10      15      20      25
```

Ketqua.Out

```
3
0  0  0  0  1  0  0
1  0  0  1  0  0  0
0  1  1  0  0  0  0
```

11. Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy viết chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N và K; hai số được viết cách nhau bởi một vài khoảng trống;
- Dòng kế tiếp ghi lại N số của dãy số A[], hai số được viết cách nhau một vài khoảng trống.

Các dãy con thoả mãn điều kiện tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ dưới đây sẽ minh hoạ cho file dayso.in và ketqua.out của bài toán.

Dayso.in	**Ketqua.out**
5 50 5 10 15 20 25	3 10 15 25 5 20 25 5 10 15 20

12. Cho dãy A[] gồm N số tự nhiên khác nhau, số tự nhiên K và số tự nhiên B. Hãy viết chương trình liệt kê tất cả các dãy con K phần tử của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng B. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại ba số tự nhiên N, K, B; Ba số được viết cách nhau bởi một vài khoảng trống;
- Dòng kế tiếp ghi lại N số của dãy số A[], hai số được viết cách nhau một vài khoảng trống.

Các dãy con K phần tử thoả mãn điều kiện tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các dãy con K phần tử có tổng các phần tử đúng bằng tìm được;

- Những dòng kế tiếp mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ dưới đây sẽ minh hoạ cho file dayso.in và ketqua.out của bài toán.

Dayso.in					**Ketqua.out**		
5	3	50			2		
5	10	15	20	25	10	15	25
					5	20	25

13. Cho dãy gồm N số nguyên phân biệt A[] = $\{a_1, a_2, .., a_N\}$ và số tự nhiên K ($K \le N \le 100$). Hãy viết chương trình liệt kê tất cả các dãy con K phần tử tăng dần của dãy số A[]. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

i. Dòng đầu tiên ghi lại hai số tự nhiên N, K. Hai số được viết cách nhau một vài khoảng trống;

ii. Những dòng kế tiếp ghi lại N số nguyên của dãy số A[], hai số khác nhau được viết cách nhau một vài khoảng trống.

Các dãy con K phần tử tăng dần của dãy số A[] tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số các dãy con K phần tử tăng dần của dãy số A[] tìm được;
- M dòng kế tiếp, mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file dayso.in dưới đây sẽ cho ta file ketqua.out tương ứng.

dayso.in					**ketqua.out**		
5	3				7		
2	5	15	10	20	2	5	15
					2	5	10
					2	5	20
					2	15	20
					2	10	20
					5	15	20
					5	10	20

14. Cho ma trận vuông $C_{i,j}$ cấp N ($1 \le i, j \le N \le 100$) gồm N^2 số tự nhiên và số tự nhiên K(*Các số không nhất thiết phải khác nhau*) ghi lại trong file matran.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N và K. Hai số được viết cách nhau một vài khoảng trống;
- N dòng kế tiếp ghi lại ma trận vuông Ci,j; Hai phần tử khác nhau của ma trận được ghi cách nhau bởi một vài khoảng trống.

Hãy viết chương trình lấy mỗi hàng, mỗi cột duy nhất một phần tử sao cho tổng các phần tử này đúng bằng K. Kết quả tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số các nghiệm tìm được của bài toán.
- Những dòng kế tiếp, mỗi dòng ghi lại N số là một phương án của bài toán, số thứ i ghi lại giá trị j tương ứng với chỉ số cột của phần tử được lựa chọn. Các số được viết cách nhau một vài khoảng trống.

Ví dụ về file viec.in và ketqua.out:

matranc.in						**ketqua.out**					
6	180					6					
10	64	57	29	18	15	2	1	4	6	3	5
34	20	19	30	16	12	3	6	1	5	4	2
57	49	40	16	11	19	3	6	2	4	5	1
29	21	46	26	21	18	4	3	2	6	1	5
28	16	11	21	21	37	5	3	2	6	1	4
15	12	15	48	37	30	6	3	2	5	1	4

14. Cho ma trận vuông C = (c_{ij}) cấp N ($1 \le i, j \le N \le 100$) gồm N^2 số tự nhiên (*các số không nhất thiết phải khác nhau*) ghi lại trong file matran.in theo khuôn dạng sau :

- Dòng đầu tiên ghi lại số tự nhiên N là cấp của ma trận vuông C;
- N dòng kế tiếp ghi lại ma trận vuông C = (c_{ij}); Hai phần tử khác nhau của ma trận được ghi cách nhau bởi một vài khoảng trống.

Hãy viết chương trình lấy trên mỗi hàng, mỗi cột duy nhất một phần tử sao cho tổng các phần tử này là lớn nhất. Kết quả tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại tổng giá trị nhỏ nhất của N phần tử tìm được;
- N dòng kế tiếp, mỗi dòng ghi lại ba số i, j, c_{ij} tương ứng với chỉ số hàng, chỉ số cột và giá trị phần tử tương ứng của ma trận. Ba số được viết cách nhau một vài khoảng trống.

Ví dụ về file viec.in và ketqua.out:

matran.in

6								
10	14	27	29	18	**27**	1	6	27
34	20	19	34	16	12	2	1	34
57	37	40	**57**	11	19	3	4	57
29	21	**46**	26	21	18	4	3	46
27	**37**	11	21	21	37	5	2	37
55	12	15	48	**37**	35	6	5	37

ketqua.out

238

Bài tập Đồ thị

1. Cho đồ thị vô hướng G =<V,E> gồm N đỉnh và M cạnh. Ta định nghĩa khuôn dạng lưu trữ đồ thị bằng ma trận kề, danh sách cạnh, danh sách kề dưới dạng file như sau:

Khuôn dạng file ma trận kề:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được ghi cách nhau bởi một vài ký tự trống.

Khuôn dạng file danh sách cạnh:

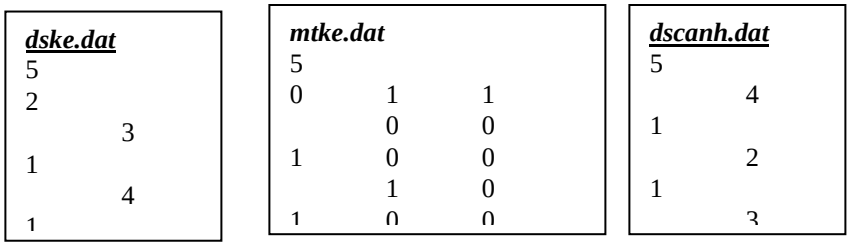
- Dòng đầu tiên ghi lại số tự nhiên N và M tương ứng với số đỉnh và số cạnh của đồ thị, hai số được ghi cách nhau bởi một vài ký tự trống;
- M dòng kế tiếp mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Khuôn dạng file danh sách kề:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh của đồ thị;
- N dòng kế tiếp mỗi dòng ghi lại danh sách kề của đỉnh tương ứng. Hai đỉnh trong cùng một danh sách kề được phân biệt với nhau bằng một hoặc vài kí tự trống, đỉnh không có cạnh nối với nó (đỉnh cô lập) được ghi giá trị 0.

Hãy viết chương trình chuyển đổi biểu diễn đồ thị giữa các file danh sách kề *(dske.dat),* ma trận kề (*mtke.dat*) và danh sách cạnh (*dscanh.dat*).

Ví dụ đồ thị gồm 5 đỉnh, 5 cạnh được biểu diễn trong các file như sau:



2. Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán BFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới danh sách kề theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, u tương ứng với số đỉnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp, mỗi dòng ghi lại danh sách kề của đỉnh tương ứng, hai đỉnh khác nhau của cùng một danh sách kề được ghi cách nhau bởi một vài ký tự trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, M tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- M dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

```
5 1
2 3 4 5

1 3 5
1 2 4
1 3 5
1 2 4
```

cay.out

```
5 4
1 2
1 3
1 4
1 5
```

3. Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán DFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới danh sách kề theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, u tương ứng với số đỉnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp, mỗi dòng ghi lại danh sách kề của đỉnh tương ứng, hai đỉnh khác nhau của cùng một danh sách kề được ghi cách nhau bởi một vài ký tự trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, M tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- M dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

```
5 1
2 3 4 5

1 3 5
1 2 4
1 3 5
1 2 4
```

cay.out

```
5 4
1 2
1 3
1 4
1 5
```

4. Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán BFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại ba số tự nhiên N, M và u tương ứng với số đỉnh, số cạnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Ba số được viết cách nhau bởi một vài khoảng trống.
- M dòng kế tiếp, mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được viết cách nhau một vài khoảng trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, K tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- K dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

```
5        8
         1
1
         2
1
         3
1
         4
```

cay.out

```
5        4
1        2
1        3
1        4
1        5
```

5. Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán DFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại ba số tự nhiên N, M và u tương ứng với số đỉnh, số cạnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Ba số được viết cách nhau bởi một vài khoảng trống.

- M dòng kế tiếp, mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được viết cách nhau một vài khoảng trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, K tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- K dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

```
5
        8
        1
1
        2
1
        3
1
        4
```

cay.out

```
5       4
1       2
2       3
3       4
4       5
```

6. Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán BFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới dạng ma trận kề theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, u tương ứng với số đỉnh và đỉnh bắt đầu xây dựng cây khung. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, K tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- K dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in	
5	
0	1
	1
	1
	1
1	0

cay.out	
5	4
1	2
1	3
1	4
1	5

A. Cây nhị phân.

1- Xây dựng các thao tác với cây nhị phân cài đặt bằng con trỏ có cấu trúc:

struct CanBo { long MaSo; char HoTen[8]; };

struct Node { CanBo Info; Node *Left, *Right; };

13. Khởi tạo cây, tạo Node lá.
14. Thêm một Node lá vào bên trái một Node.
15. Thêm một Node lá vào bên phải một Node.
16. Xóa một Node lá.
17. Xóa cả cây.
18. Duyệt cây theo thứ tự trước
19. Duyệt cây theo thứ tự giữa
20. Duyệt cây theo thứ tự sau.

2- Xây dựng các thao tác với cây nhị phân tìm kiếm cài đặt bằng con trỏ có cấu trúc:

struct SinhVien { long MaSo; char HoTen[8]; };

struct Node { SinhVien Info; Node *Left, *Right; };

1. Khởi tạo cây, tạo Node lá.
2. Thêm một Node lá
3. Xóa một Node.
4. Xóa cả cây.
5. Tìm kiếm nhị phân theo MaSo của SinhVien
6. Duyệt cây theo thứ tự trước

7. Duyệt cây theo thứ tự giữa
8. Duyệt cây theo thứ tự sau