Họ và tên: Phạm Thành An

Lớp: CNTT3-K63

Mã sinh viên: 221230727

Lớp học phần: N03

**BÁO CÁO BÀI THI**

- Số bài tập đã làm 22/23

- Số bài tập chưa làm 1/23 (bài số 23)

**Bài 1:**

\* Class IndexMaxPQ bao gồm:

- Các thuộc tính:

+ private int n: số lượng phần tử có trong PriorityQueue.

+ private int[] pq: dùng để lưu chỉ mục của một phần tử.

+ private int[] qp: dùng để lưu nghịch đảo của pq chỉ ra là phần tử thứ mấy trong PriorityQueue.

+ private Key[] keys: dùng để lưu giá trị của key ở chỉ mục i.

- Các phương thức:

+ public boolean isEmpty(): kiểm tra xem PriorityQueue có rỗng không.

+ public boolean contains(int i): kiểm tra xem PriorityQueue có chứa chỉ mục i không.

+ public int size(): trả về kích thước của hàng đợi ưu tiên.

+ public void insert(int i, Key key): chèn một phần tử có chỉ mục i và có giá trị key vào hàng đợi ưu tiên.

+ private void swim(int k): giúp xây dựng một cấu trúc max heap (phần tử cha luôn lớn hơn phần tử con).

+ private void exch(int i, int j): giúp hoán vị hai phần tử có chỉ mục i và j.

+ private boolean less(int i, int j): giúp so sánh hai key ở hai chỉ mục i và j.

+ public int maxIndex(): trả về chỉ mục với khóa là lớn nhất.

+ public Key maxKey(): trả về key lớn nhất.

+ public int delMax(): xóa đi khóa lớn nhất, trả về chỉ mục của khóa lớn nhất đó.

+ private void sink(int k): giúp xây dựng một cấu trúc max heap (phần tử cha luôn lớn hơn phần tử con).

+ public Key keyOf(int i): trả về giá trị của key với chỉ mục i.

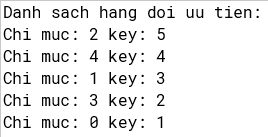
+ public void changeKey(int i, Key key): thay đổi giá trị của key tại chỉ mục i.

+ public void increaseKey(int i, Key key): tăng key được liên kết với chỉ mục i lên giá trị key được chỉ định.

+ public void decreaseKey(int i, Key key): giảm key được liên kết với chỉ mục i lên giá trị key được chỉ định.

+ public void delete(int i): xóa key trên hàng đợi ưu tiên được liên kết với chỉ mục i.

\* Kết quả:



**Bài 9:**

\* Class FileFrequencyIndex bao gồm:

- Các thuộc tính:

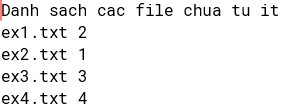
+ private ST<String, ST<File,Integer>> st: là một simple table cho biết với một từ thì có những file nào chứa từ đó và file đó chứa từ số lượng từ đó là bao nhiêu.

- Các phương thức:

+ public void addFile(File file): thêm một file vào trong.

+ public void query(String word): truy xuất danh sách các file chứa từ đó cùng với đó là in ra trong file đó chứa từ đó bao nhiêu lần.

\* Kết quả:



**Bài 20:**

\* Class Job bao gồm:

- Các thuộc tính:

+ int start: thời gian bắt đầu của một công việc.

+ int finish: thời gian kết thúc của một công việc.

- Các phương thức:

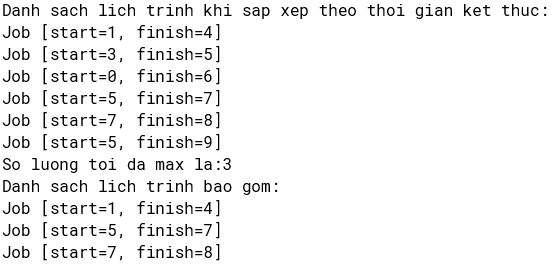
+ public int compareTo (Job y): phương thức giúp so sánh hai công việc với nhau theo thời gian kết thúc.

+ public int getStart (): lấy thời gian bắt đầu của một công việc.

+ public int getFinish (): lấy thời gian kết thúc của một công việc.

+ public String toString(): in ra một công việc.

\* Kết quả:



**Bài 21:**

\* Class Job bao gồm:

- Các thuộc tính:

+ int start: thời gian bắt đầu của một công việc.

+ int finish: thời gian kết thúc của một công việc.

+ int profit: lợi nhuận của một công việc.

- Các phương thức:

+ public int compareTo (Job y): phương thức giúp so sánh hai công việc với nhau theo thời gian kết thúc.

+ public int getProfit(): lấy lợi nhuận của một công việc.

+ public String toString(): in ra một công việc.

+ public static int findLastNonConflictingJob(List<Job> jobs, int n): tìm ra vị trí cuối cùng của công việc không bị xung đột với công việc ở vị trí n.

+ public static int maxProfit(List<Job> jobs,ArrayList<Job>[] ans): tìm ra lợi nhuận lớn nhất và cho biết nên làm những công việc nào để đạt được lợi nhuận lớn nhất.

\* Kết quả:

