NAME: Afnan Syed
UFID: 88459318
EMAIL: afnansyed@ufl.edu

LANGUAGE: C++

CLASS:

- flightRequest
  - Initialized as described in the project description file.
  - Represents a flight
  - int flightID, airlineID, submitTime, priority, duration, runwayID, startTime, runway, endTime, lifecycl;
  - All of them are initialized to -1 except lifecycl which is 0. lifecycl can have 4 values: 0 is pending, 1 is scheduled, 2 is in progress, 3 is landing

GLOBAL VARIABLES:

vector<flightRequest> flights : all flights
vector<int> runways : all runway ids
vector<int> runfree : stores time when runway (referring to runways vector) is free
int rwyct : keep track of runwayids added
int currentTime : time value the currently considering

HEAP FUNCTIONS:

- **void push(int f, int p, int s)** : add a flight
- **int pop()** : remove root from heap
- **void incr(int f, int newp)** : change flight priority
- **void remove(int f)** : remove node
- **void clearh()** : reset all values in heap
- **bool goup(int a, int b)** : compares node a and node b and checks if node a should be pushed up
- **int combine(int a, int b)** : combines two heaps
- **int mergeall(int first)** : merge all heaps after root is deleted

The integer parameters are the index of a node in the heap arrays.
f = flightID, p = priority, s = submitTime

MAX PAIRING HEAP IMPLEMENTATION:

- I used array based implementation for heap.

vector<int> hpri : priority for a node
vector<int> hsub : submit time for a node
vector<int> hid : flight id for a node

vector<int> hpar : parent index
vector<int> hleft : stores left child index of parent
vector<int> hnext : stores next sibling index of left child

int hroot : root node
int hsz : index wher node is empty
vector<int> idtoh : stores heap index of flight ids

SCHEDULING FUCTIONS:

- **void settle()** : settle flight completions
- **void lifenext()** : check if flight needs to be moved to in progress

RESEDUALING:

- **void resechdule(bool print) :**
  Find the flights that have not been scheduled by looping through the flights vector, then cleans the existing heap.
  Flights that have not started get put on the heap.
  Runway is reset, the in progress runways are put in, greedy algorithm for scheduling (highest priority first, then smaller runway id

TIME ADVANCES:

- **void timechang(int newt)** : update when current time changes
  Updates current time to the new one, calles settle(), lifenext(), resechdule()

OPERATIONS (implemented as described in the project description file)

1. **void Initialize(int runwayCount)**
2. **void SubmitFlight(int flightID, int airlineID, int submitTime, int priority, int duration)**
3. **void CancelFlight(int flightID, int currentTime)**
4. **void Reprioritize(int flightID, int currentTime, int newPriority)**
5. **void AddRunways(int count, int currentTime)**
6. **void GroundHold(int airlineLow, int airlineHigh, int currentTime)**
7. **void PrintActive()**
   void PrintSchedule(int t1, int t2)
8. **void Tick(int t)**

MAIN PROGRAM:

1. The input file is opened.
2. Next the output file is created after parsing through the name of the input file.
3. The heap arrays are initialized.
4. I then have a while loop to read each line of the input file.
    a. The line is first parsed to get the command which are characters till the open parentheses. The rest of it is stored in a string.
    b. This string which is to get the numbers is parsed by saving the numbers into a vector each time a comma is there.
    c. The last number before the close parentheses is then stored.
5. Then based on the command, one of the Operations commands is called.

STRUCTURE OF PROGRAM:

1. **main()**
    a. Reading input file
    b. Parsing command
    c. Calling operation function based on command
2. **Flow of most operations**
    a. timechang(int newt)
    b. settle()
    c. lifenext()
    d. exe operation
    e. resechdule(bool print)
3. **Heap implementation**
    a. push()
        i. push()
        ii. combine()
        iii. goup() : compare root node to see if update
    b. pop()
        i. pop()
        ii. mergeall()
        iii. combine()
        iv. goup() : compare root node to see if update
    c. incr() : increase-key
        i. Remove node
        ii. combine() using the root node
    d. remove()
        i. Remove node
        ii. combine() using the root node
        iii. pop()

references used to understand concepts to write the code:

https://www.geeksforgeeks.org/cpp/priority-queue-of-pairs-in-c-ordered-by-first/

https://www.geeksforgeeks.org/dsa/pairing-heap/

https://www.geeksforgeeks.org/dsa/bubble-sort-algorithm/

https://www.geeksforgeeks.org/dsa/selection-sort-algorithm-2/

Used project description file to write some of the comments for functions in the project main.cpp file.