Report

Name: venkata sai sri krishna
ufid: 4891-9465
uf email: ravipati.v@ufl.edu

Function prototypes

Building

```
int id,exec_time,total_time;

Building(int _id,int _exec_time, int _total_time);
int get_id();
int get_exec();
int get_total();
void inspect();
void increment();
void update(int new_exec);
```

IO

```
string file_name;

vector<vector<string>> read_input();
vector<string> read_line(ifstream& file);
vector<string> parse_list(string s);
```

heap_node

```
Building building;
rb_node* map_ptr;
```

min_heap

```
int size;
vector<heap_node> data;

void heapify(int index);
void insert(heap_node);
void pop();
bool compare(int p,int i);
heap_node get_min();
int left(int index);
int right(int index);
int parent(int index);
void inspect();
bool empty();
```

rb_node

```
Building building;
```

```
    COLOR color;
    rb_node *left, *right, *parent;

    void to_black();
    void to_red();
    bool is_black(rb_node* node);
    bool is_red(rb_node* node);
    rb_node* get_sib();
    rb_node* get_uncle();
```

red black tree

```
    rb_node *root;
    int size;

    rb_node* insert(Building building);
    void erase(int id);
    void erase(rb_node *root,int id);
    void replace_in_parent(rb_node* root,rb_node* new_node);
    rb_node *find(int id);
    rb_node* find_min(rb_node*);
    void increment(int id);
    bool empty();
    void update(rb_node* node,int new_exec);
    void insert_fix(rb_node* node);
    void ll_rotate(rb_node* node);
    void rr_rotate(rb_node* node);
    void lr_rotate(rb_node* node);
    void rl_rotate(rb_node* node);
    void insert_rotation_fix(rb_node* node);
    void range_print(int l,int r,ofstream& out);
    void range_print_util(rb_node* root,int l,int r,vector<Building>& res,ofstream& out);
    void erase_fix(rb_node* y,rb_node* py);
    int red_children(rb_node*);
    void print(int id,ofstream& out);
    void print_util(rb_node* root,int id,ofstream& out);
```

wanye

```
    min_heap heap;
    rb_tree map;
    int time;
    vector<vector<string>> commands;

    wanye(vector<vector<string>>commands);
    heap_node select_building();
    void build_city();
    void insert(Building b);
    void remove_from_map(heap_node node);
    void perform_input_op();
    bool empty();
```

Building

This is the Building class which stores id,execution time and total time needed.