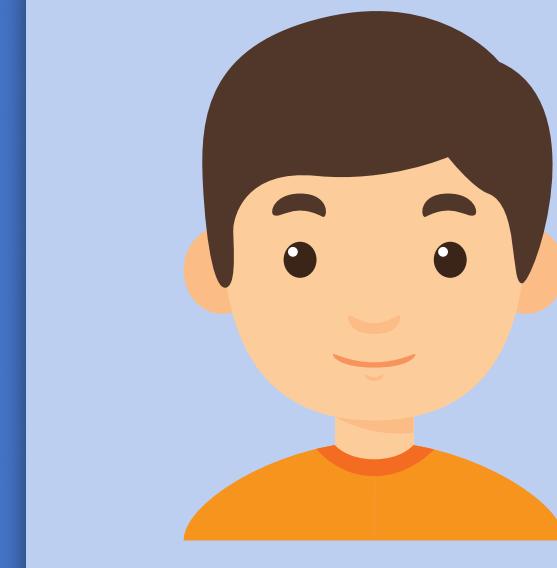


# Edward



Marozi.t@gmail.com

```
179 void* list_pop_front(list_t* row_list);
180 void list_process_all(l_node_t* p, void (*process)(void*));
181 void list_free(list_t* list);
182 void create_linked_list(int num_center, row_t* center_points, list_t* row_list);
183 void linear_list_search(FILE* task3, liHOME_lisPROJECTS_st);
184 /* Binary Tree */
185 int make_unique(double* array, int n);
186 void no_free(void* v);
187 void perfect_insert(bst_flux_t* bst_flux, double* array, int low, int high);
188 int bst_flux_insert(bst_flux_t* bst_flux, void* d);
189 void bst_flux_free(bst_flux_t* bst_flux);
190 /* bst_flux_free_subtree(bst_flux_t* bst_flux, bst_flux_node_t* n);
191 bst_t* bst_flux_new(void (*delfunc)(void*));
192 */
193 /* Search */
194 /* binary_search(double arr[], int l, int r, double x, FILE* task3);
195 binary search on array */
196 /* bst_node_t* bst_flux_find(bst_flux_t* bst_flux, double d, FILE* task3);
197 */
198 /* four functions */
199 calc_n_m(FILE* input_file, grid_t* grid);
200
201 **** Task one: Calculate the maximum flux difference in the data set ****
202 ****
203
204 void maxfluxdiff(const char* flow_file)
205 {
206     FILE *input_file;
207     row_t *max_flux_arr = [0];
208     char *headers;
209
210     /* Dynamically assigns the memory for the first row */
211     headers = (char*)malloc(HEADER_CHARS * sizeof(char));
212 }
```

Marozzi

## ABOUT ME

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO EIUSMOD TEMPOR INCIDIDUNT UT LABORE ET DOLORE MAGNA ALIQUA. UT ENIM AD MINIM VENIAM, QUIS NOSTRUD EXERCITATION ULLAMCO LABORIS NISI UT ALIQUIP EX EA COMMODO CONSEQUAT. DUIS AUTE IRURE DOLOR IN REPREHENDERIT IN VOLUNTATE VELIT ESSE CILLUM DOLORE



## My Work

[SEE ALL PROJECTS](#)

