Găujăneanu Nicoleta Monica
Grupa 414D

Lucrare 2 SDA

Exercitiul 1

a)

Pseudocod

se defineste functia **cmp**, ca functie de comparare cu parametrii **a** si **b**:

```
daca a = b atunci
        returneaza 0
daca a < b atunci
        returneaza -1
daca a > b atunci
        returneaza 1
altfel
        returneaza 0
```

se defineste functia **selectionSort**, ca functie de sortare cu parametrii **nodes**, **first**, **last** si functia **cmp** descrisa mai sus:

se declara o variabila locala auxiliara pe care o vom folosi la sortare, in care vom stoca prima valoare. Se va numi **aux**

se declara o variabila locala pe care o vom initializa la fiecare iteratie cu cate un element din vectorul nodes, incepand cu primul element din vector. Se va numi **bestValue**

se declara o variabila locala pe care o vom initializa la fiecare iteratie cu pozitia elementului la care suntem, element pe care il stocam in variabila amintita anterior. Se va numi **bestI**

cat timp first < last executa

bestValue = nodes[first]

bestI = first

pentru i = first + 1, i <= last, i = i + 1 executa

daca cmp(nodes[i], bestValue) < 0 atunci

bestValue = nodes[i]

bestI = i

la acest pas se va face sortarea propriu-zisa in care vom interschimba cele 2 valori din vector, nodes[bestI] cu nodes[first]:

aux = nodes[bestI]

nodes[bestI] = nodes[first]

nodes[first] = aux

incrementam first cu 1 pentru a putea trece la urmatorul element din vector: first = first + 1

pentru i = 0, i <= 5, i = i + 1 executa

afiseaza vectorul sortat: afiseaza nodes[i]

Cod

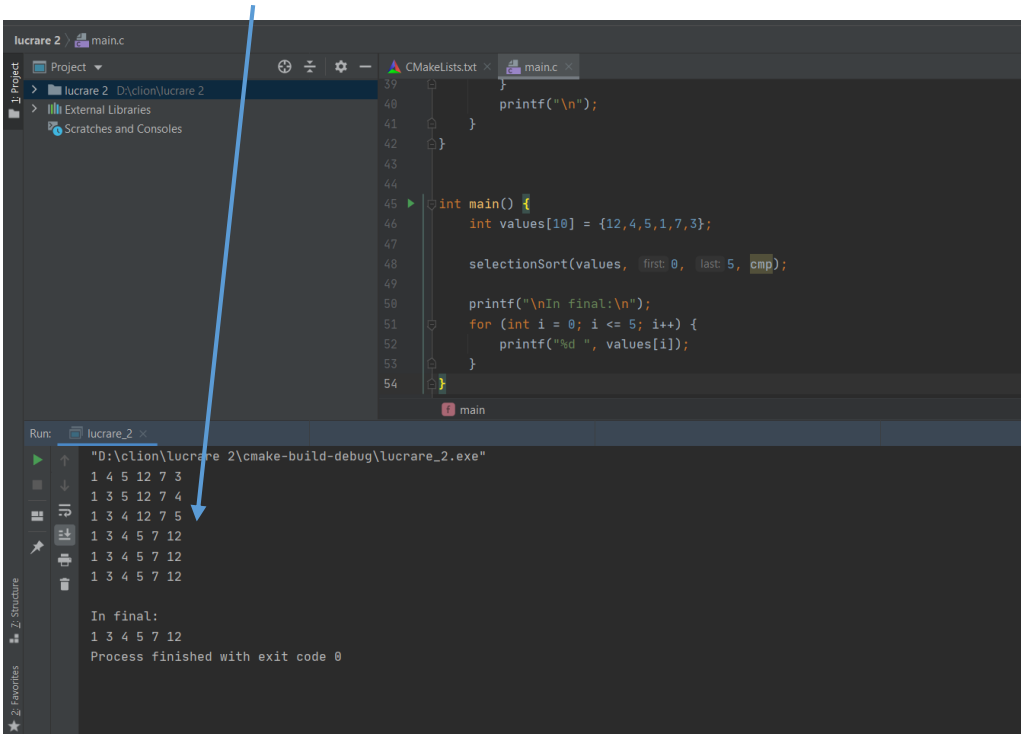
```
#include <stdlib.h>
#include <stdio.h>

int cmp(int a, int b) {
	if (a == b) {
		return 0;
	}
```

```
	if (a < b) {
		return -1;
	}
	if (a > b) {
		return 1;
	}

	return 0;
}



// Vom sorta in intervalul [first, last]
void selectionSort(int nodes[], int first, int last, int (*cmp)(int *a, int *b)) {
	int bestValue, bestI, aux;
	while (first <= last) {
		bestValue = nodes[first];
		bestI = first;

		for (int i = first + 1; i <= last; i++) {
			if (cmp(nodes[i], bestValue) < 0) {
				bestValue = nodes[i];
				bestI = i;
			}
		}
		aux = nodes[bestI];
		nodes[bestI] = nodes[first];
		nodes[first] = aux;
```

```
            first++;
            for (int i = 0; i <= last; i++) {
                printf("%d ", nodes[i]);
            }
            printf("\n");
        }
}

//Pentru a verifica daca functioneaza
int main() {
        int values[10] = {12,4,5,1,7,3};

        selectionSort(values, 0, 5, cmp);

        printf("\nIn final:\n");
        for (int i = 0; i <= 5; i++) {
            printf("%d ", values[i]);
        }
}
```

Găujăneanu Nicoleta Monica
Grupa 414D

Sortarea pas cu pas



b)

```
struct node_btree {
        int value;
        struct node_btree *left, *right;
};

struct queue_node {
        struct node_btree *b_node;
        struct queue_node *next;
};
```

```
void destroy_queue(struct queue_node *node) {
	if (node == NULL) {
		return;
	}
	destroy_queue(node->next);
	free(node);
}

// inseram un nou nod la sfarsitul cozii
void insert_in_queue(struct queue_node **q_node, struct node_btree *b_node) {
	struct queue_node *new_q_node = malloc(sizeof(struct queue_node));
	new_q_node->next = NULL;
	new_q_node->b_node = b_node;

	if (*q_node == NULL) {
		*q_node = new_q_node;
	} else {
		struct queue_node *node = *q_node;
		while (node->next != NULL) {
			node = node->next;
		}
		node->next = new_q_node;
	}
}

struct node_btree * pop_from_queue(struct queue_node **q_node) {
	if (*q_node == NULL) {
```

```
		return NULL;
	}

	struct node_btree *b_node = (*q_node)->b_node;
	struct queue_node *first_q_node = *q_node;
	*q_node = (*q_node)->next;
	free(first_q_node);

	return b_node;
}

struct node_btree *create_node(int value) {
	struct node_btree *new_node = malloc(sizeof(struct node_btree));

	new_node->value = value;
	new_node->left = NULL;
	new_node->right = NULL;

	return new_node;
}

void destroy_btree(struct node_btree *node) {
	if (node == NULL) {
		return;
	}
	destroy_btree(node->left);
	destroy_btree(node->right);
```

```
	free(node);
}

// traversarea in latime a arborelui binar
void tl_tree(struct node_btree *r) {
	if (r == NULL) {
		return;
	}

	struct queue_node *queue = NULL;
	insert_in_queue(&queue, r);

	while (queue != NULL) {
		struct node_btree *current_b_node = pop_from_queue(&queue);
		printf("%d ", current_b_node->value);
		if (current_b_node->left != NULL) {
			insert_in_queue(&queue, current_b_node->left);
		}
		if (current_b_node->right != NULL) {
			insert_in_queue(&queue, current_b_node->right);
		}
	}
}
```

Găujăneanu Nicoleta Monica
Grupa 414D

```
//Pentru verificare am facut si main ul
int main() {

    struct node_btree *root = create_node(1);
    root->left = create_node(2);
    root->left->left = create_node(5);
    root->right = create_node(3);
    root->right->right = create_node(4);
    tl_tree(root);
}
```

Arborele traversat in latime

Găujăneanu Nicoleta Monica
Grupa 414D

Exercitiul 2

```
#include <stdlib.h>

#include <stdio.h>

#include <string.h>


int cmp(int a, int b) {

	if (a == b) {

		return 0;

	}

	if (a < b) {

		return -1;

	}

	if (a > b) {

		return 1;

	}


	return 0;

}


// Vom sorta in intervalul [first, last]

void selectionSort(int nodes[], int first, int last, int (*cmp)(int *a, int *b)) {

	int bestValue, bestI, aux;


	while (first <= last) {

		bestValue = nodes[first];

		bestI = first;
```

```
		for (int i = first + 1; i <= last; i++) {
			if (cmp(nodes[i], bestValue) < 0) {
				bestValue = nodes[i];
				bestI = i;
			}
		}

		aux = nodes[bestI];
		nodes[bestI] = nodes[first];
		nodes[first] = aux;

		first++;
	}
}

struct node_btree {
	int value;
	char letter;
	struct node_btree *left, *right;
};

struct queue_node {
	struct node_btree *b_node;
	struct queue_node *next;
};

void destroy_queue(struct queue_node *node) {
```

```
	if (node == NULL) {
		return;
	}
	destroy_queue(node->next);
	free(node);
}

// inseram un nou nod la sfarsitul cozii
void insert_in_queue(struct queue_node **q_node, struct node_btree *b_node) {
	struct queue_node *new_q_node = malloc(sizeof(struct queue_node));
	new_q_node->next = NULL;
	new_q_node->b_node = b_node;

	if (*q_node == NULL) {
		*q_node = new_q_node;
	} else {
		struct queue_node *node = *q_node;
		while (node->next != NULL) {
			node = node->next;
		}
		node->next = new_q_node;
	}
}

struct node_btree * pop_from_queue(struct queue_node **q_node) {
	if (*q_node == NULL) {
		return NULL;
```

```
        }

        struct node_btree *b_node = (*q_node)->b_node;
        struct queue_node *first_q_node = *q_node;
        *q_node = (*q_node)->next;
        free(first_q_node);

        return b_node;
}

struct node_btree *create_node(int value, char letter) {
        struct node_btree *new_node = malloc(sizeof(struct node_btree));

        new_node->value = value;
        new_node->letter = letter;
        new_node->left = NULL;
        new_node->right = NULL;

        return new_node;
}

void destroy_btree(struct node_btree *node) {
        if (node == NULL) {
                return;
        }
        destroy_btree(node->left);
        destroy_btree(node->right);
```

```
	free(node);
}
// traversarea in latime a arborelui binar
void tl_tree(struct node_btree *r) {
	if (r == NULL) {
		return;
	}

	struct queue_node *queue = NULL;
	insert_in_queue(&queue, r);

	while (queue != NULL) {
		struct node_btree *current_b_node = pop_from_queue(&queue);
		printf("%c:%d ", current_b_node->letter, current_b_node->value);
		if (current_b_node->left != NULL) {
			insert_in_queue(&queue, current_b_node->left);
		}
		if (current_b_node->right != NULL) {
			insert_in_queue(&queue, current_b_node->right);
		}
	}
}
int cmp_nodes(struct node_btree *n1, struct node_btree *n2) {
	if (n1->value == n2->value) {
		if (n1->letter == n2->letter) {
			return 0;
		}
```

```
		if (n1->letter < n2->letter) {
			return -1;
		} else {
			return 1;
		}
	}
	if (n1->value < n2->value) {
		return -1;
	} else {
		return 1;
	}
}

// Vom sorta in intervalul [first, last]
void selectionSortNodes(struct node_btree *nodes[], int first, int last,
		int (*cmp_nodes)(struct node_btree *n1, struct node_btree *n2)) {

	int bestI;
	struct node_btree *bestNode, *aux;

	while (first <= last) {
		bestNode = nodes[first];
		bestI = first;
		for (int i = first + 1; i <= last; i++) {
			if (cmp_nodes(nodes[i], bestNode) < 0) {
				bestNode = nodes[i];
				bestI = i;
```

```
                }
            }

            aux = nodes[bestI];
            nodes[bestI] = nodes[first];
            nodes[first] = aux;

            first++;
        }
    }

    int main() {
        char str[] = "afostodatacaniciodataunimparatsioimparateasa";
        int freq[26] = {0};

        for (int i = 0; i < strlen(str); i++) {
            freq[str[i] - 'a']++;
        }

    //  Cream noduri din fiecare frecventa. Literele care nu apar primesc
    //  valori foarte mari pentru a fi ignorate la sortare
        struct node_btree *nodes[26];
        int number_of_nodes = 0;
        for (int i = 0; i < 26; i++) {
            if (freq[i] == 0) {
                nodes[i] = create_node(10000000, 'a' + i);  // este doar pentru
noduri care nu ar trebui sa existe
```

```
                } else {
                        nodes[i] = create_node(freq[i], 'a' + i);
                        number_of_nodes++;
                }
        }
//      Sortam nodurile create pentru a avea doar nodurile care apar in text
        selectionSortNodes(nodes, 0, 25, cmp_nodes);
        while (number_of_nodes > 1) {
                selectionSortNodes(nodes, 0, number_of_nodes, cmp_nodes);

                struct node_btree *new_node = create_node(nodes[0]->value +
nodes[1]->value, '_');
                new_node->left = nodes[0];
                new_node->right = nodes[1];

//              Inlocuim nodurile folosite cu noduri cu valori mari pentru a le ignora ca
mai sus
                nodes[0] = create_node(10000000, ' ');
                nodes[1] = create_node(10000000, ' ');

//              Adaugam noul nod in vectorul de noduri
                nodes[0] = new_node;

                number_of_nodes--;
        }

        tl_tree(nodes[0]);
```

Găujăneanu Nicoleta Monica
Grupa 414D

```
        return 0;
}
```