Report, assignment 1
David Otgonsuren Rico

Node structure:
A node consists of a integers **cost_f, cost_g** a vectors of integers **operators_used** and an ordered vector of integeres **facts.**
cost_f - represents the value of g+h, where g is the cost to get to this node and h represents the heuristic value of the h_max function
cost_g – cost to get to the node from initial node
operators_used – a list of operators that have been used to get from initial state to this state
facts - an ordered list of ids of facts from the STRIPS representation, why it must be ordered will be explained later

After initializing the initial and goal states the A* alg is launched
A* algorithm
Input: strips representation, initial node, goal node
Output: res node
My open list is a priority queue of nodes, compared by their **cost_f** where at the top of the queue is a node with the highest **cost_f**
My closed list and list of distances that some node has reached, I have them in one strucure and it is a map where a key is **facts** and the value is the best **cost_g** so far. This is why **facts** have to be ordered. Otherwise the same state could be mapped to different values. Two vectors are equal only if they have the same size and every element in i-th position is equal. The two structures can be merged because if we visit a node we potentially put it in closed list and update the **cost_g**.

Push initial node to open list
While open list not empty
    get curr_nd from top of open list
    if curr_nd is not in closed list or the **cost_g** of curr_nd is better then previous
        update closed list with with the **cost_g**

        check if we reached goal, meaning if every fact in goal is in the curr_nd
        if yes then return curr_nd

        for every operator try to apply to curr_nd, needs to have all the preconditions
            by applying the operator to curr_nd we get a successor
            //after adding add_eff and deleting del_eff I sort the vector of facts for closed list
            cost_g of successor is set as curr_nd **cost_g** + operator cost
            put operator into operators_used of successor

            if h_max of succ != infinity(max value of integer)
                succ **cost_f** = succ **cost_g** + succ h_max

h max heuristic
Input: strips representation, current node, goal node
Output: h max value for node
implemented from notes on classical planning

the infinity value is maximum value of the integer
deltas is a vector of integers of size number of facts, initialized to infinity

```
Us is a vector of integers of size number of operators
C is an unordered set

for every fact in current node
	deltas[fact] = 0

for every operator in operators
	Us[operator] = operator.pre_size
	if operator.pre_size == 0
		for add_eff of operator
			deltas[add_eff] = min(deltas[add_eff], operator cost)

while all facts og goal node not in C
	k = arg min across deltas that are not in C
	if deltas[k] == infinity return infinity //important but not in notes of classical planning

	insert k in C
	for every operator
		for every precondition in operator
			if k precondition and Us[operator] > 0
				Us[operator] = Us[operator] – 1
				if Us[operator] == 0
					for every add_eff in operator
						deltas[add_eff] = min(deltas[add_eff],opcost+deltas[k])

h_max = max deltas[facts in goal]
```

after A* finishes and returns the res node we recereate the path and output the result
optimal cost is the **cost_g** of the resulting node
h max for init node is the **cost_f** of init node because **cost_g** for init is 0
then we iterate through operators_used of res node and add them to the output

This concludes the report.