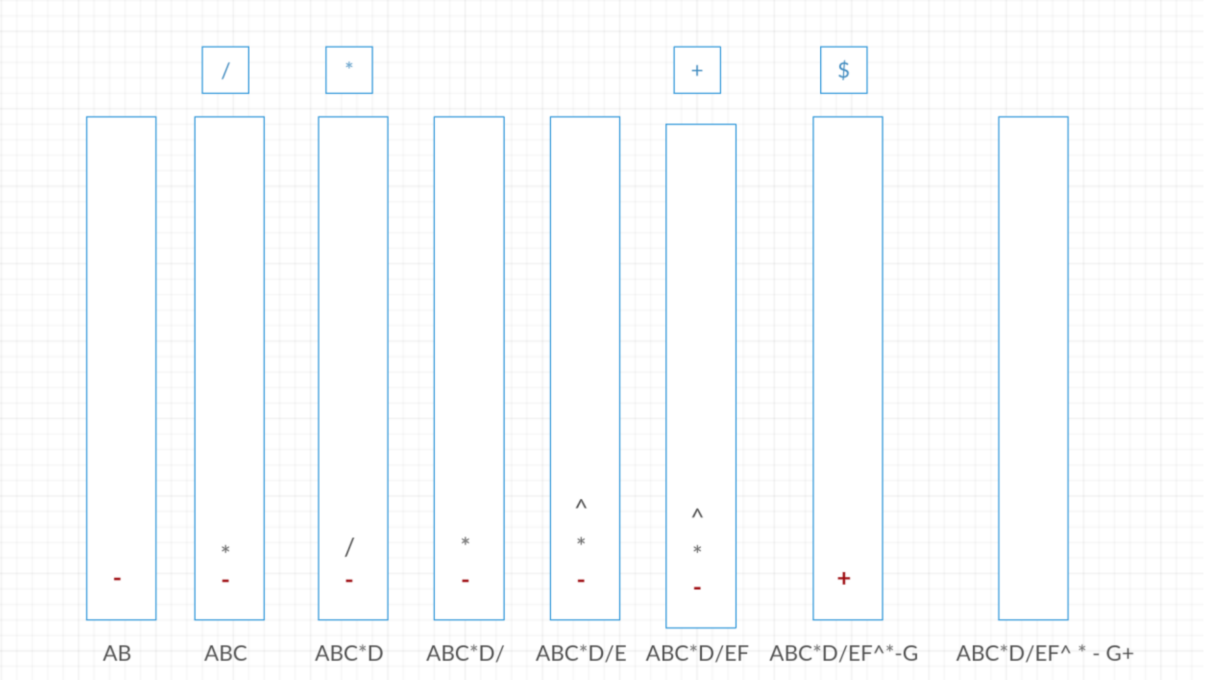
HOMEWORK3

RAHAF ALOMAR - 435201926

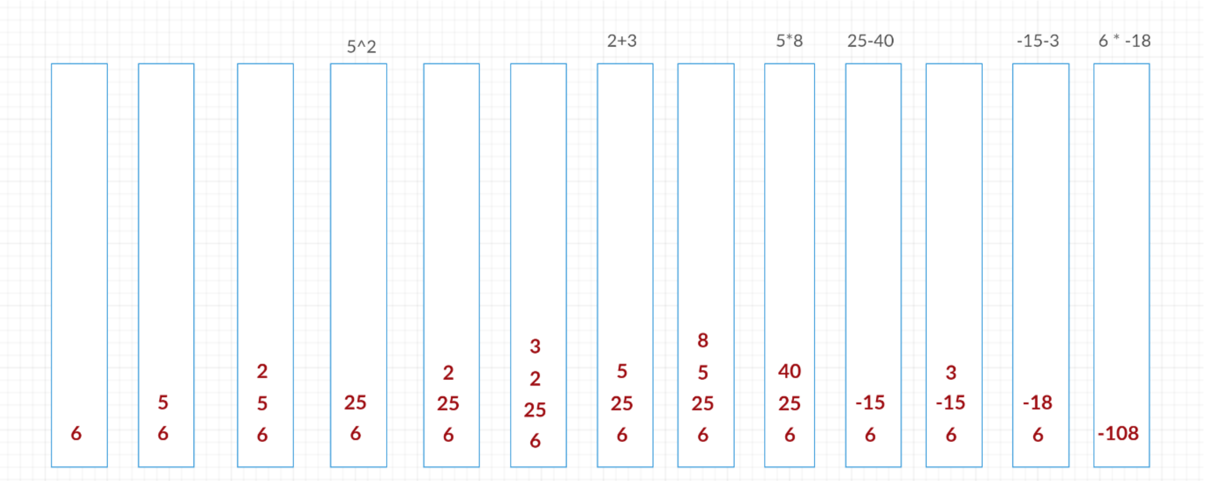
PROBLEM1:

1.1:



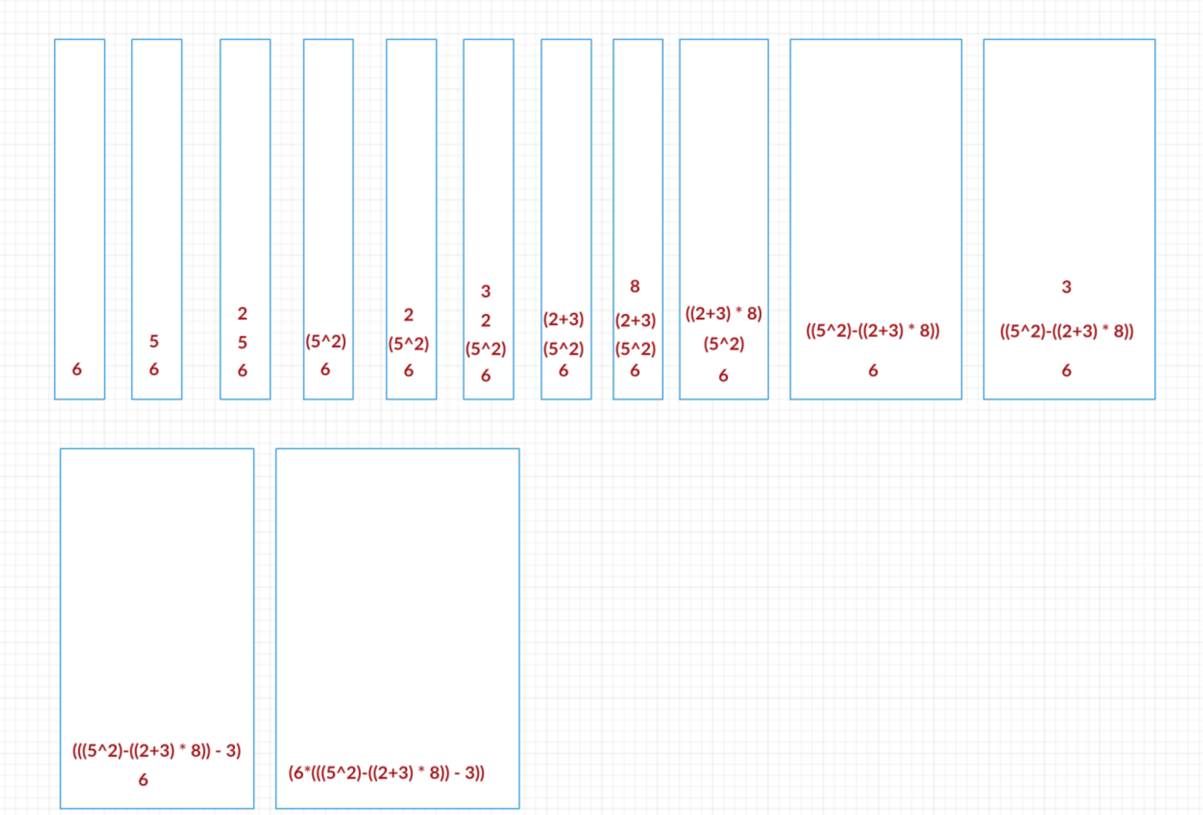
Postfix notation: A B C * D / E F ^ * – G +

1.2:



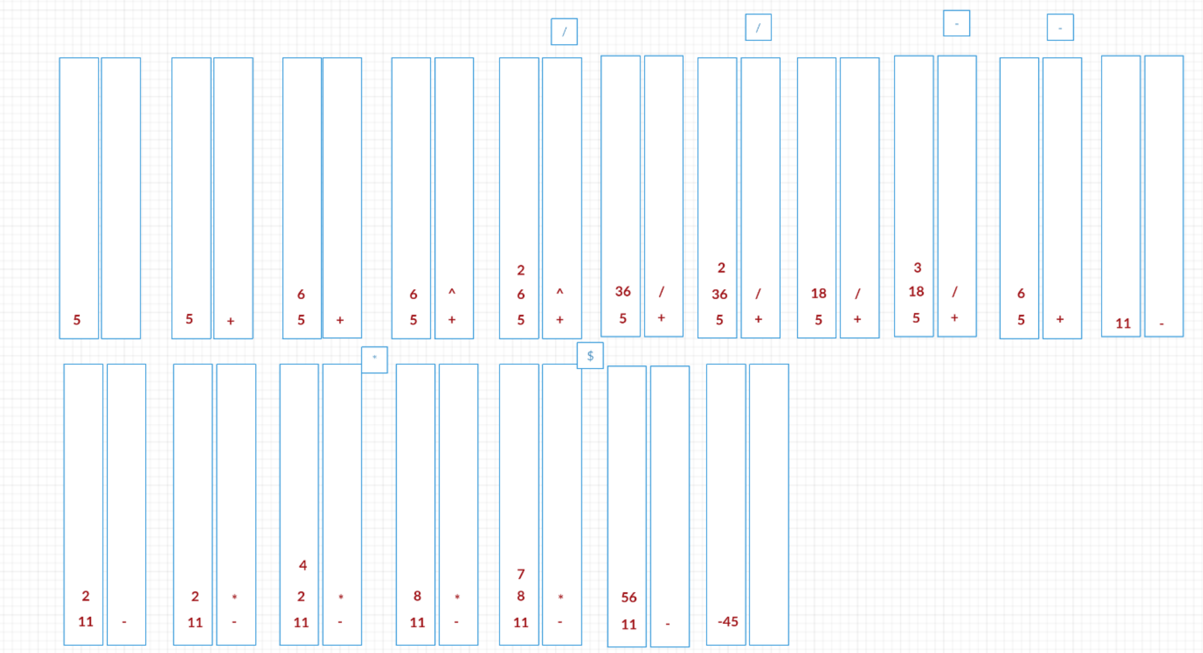
6 5 2 ^ 2 3 + 8 * – 3 – * = – 108

1.3:



INFIX NOTATION: (6 * (((5 ^ 2) – ((2+3) *8)) – 3))

1.4:



5 + 6 ^ 2 / 2 / 3 - 2 * 4 * 7 = - 45

PROBLEM2:

2.1:

```
public static <T> void removeLast(Stack<T> st) {

    Stack<T> tmp = new Stack<T>();
    while (!st.empty()) {
        tmp.push(st.pop());
    }

    if (!tmp.empty())
        tmp.pop();
    while (!tmp.empty())
        st.push(tmp.pop());

}
```

2.2:

```
public static <T> boolean topEqualsBottom(Stack<T> st) {
    if (st.empty())
        return true;

    Stack<T> tmp = new Stack<T>();
    T top = st.pop();
    tmp.push(top);
    T bottom = null;
    while (!st.empty()) {
        bottom = st.pop();
        tmp.push(bottom);
    }

    while (!tmp.empty())
        st.push(tmp.pop());

    return top.equals(bottom);
}
```

PRONLEM3:

3.1:

```
public static boolean containsMult3(int[] list, int index) {
        if (index + 1 == list.length)
            return (list[index] % 3 == 0);

        if (list[index] % 3 == 0)
            return true;

        return containsMult3(list, index + 1);
    }
```

3.2:

```
        public static boolean sameSign(int[] list, int index) {
                if (index + 1 == list.length)
                        return list[index] != 0;

                if ((list[index] > 0 && list[index + 1] < 0) || (list[index] < 0 &&
list[index + 1] > 0) || list[index] == 0)
                        return false;

                return sameSign(list, index + 1);


        }
```

PROBLEM4:

4.1:

```
        public boolean recSearch(T k) {
                return recSearch(k, head);
        }

        private boolean recSearch(T k, Node<T> tmp) {
                if (tmp == null)
                        return false;

                if (tmp.data.equals(k))
                        return true;

                return recSearch(k, tmp.next);

        }
```

4.2:

```
        public void reverse(){
                reverse(0);
        }
        private void reverse ( int index){
                if (index == top/2)
                        return;
                T tmp = nodes[index];
                nodes[index] = nodes[top-index];
                nodes[top-index]= tmp;

                reverse(index+1);

        }
```

PROBLEM5:

5.1:

```
public <T> void InsertAtBottom(Stack<T> st, T e) {
             if (st.empty()) {
                    st.push(e);
                    return;
             }

             T tmp = st.pop();
             InsertAtBottom(st, e);
             st.push(tmp);
       }
```

5.2:

```
public <T> void reverse(Queue<T> q) {
             if (q.length() == 0)
                    return;
             T tmp = q.serve();
             reverse(q);
             q.enqueue(tmp);
       }
```

5.3:

```
public <T> Queue<T> merge(Queue<T> q1, Queue<T> q2) {
             return recMerge(q1, q2, new Queue<T>());
       }

       public <T> Queue<T> recMerge(Queue<T> q1, Queue<T> q2,Queue<T> q) {
             if (q1.length() == 0 && q2.length() == 0)
                    return q;
             if (q1.length() != 0)
                    q.enqueue(q1.serve());
             if (q2.length() != 0)
                    q.enqueue(q2.serve());

             return recMerge(q1, q2, q);


       }
```

5.4:

```
public <T> Queue<T> merge2(Queue<T> q1, Queue<T> q2) {
		return recMerge2(q1, q2, new Queue<T>(), 0, 0);
	}

	public <T> Queue<T> recMerge2(Queue<T> q1, Queue<T> q2,Queue<T> q, int i, 
int j) {
		if (q1.length() == i && q2.length() == j)
			return q;
		T tmp1 = null, tmp2 = null;
		if (q1.length() != i) {
			tmp1 = q1.serve();
			q.enqueue(tmp1);
			q1.enqueue(tmp1);
			i++;
		}

		if (q2.length() != j) {
			tmp2 = q2.serve();
			q.enqueue(tmp2);
			q2.enqueue(tmp2);
			j++;
		}
		return recMerge2(q1, q2, q, i, j);

	}
```