```
Soluzione appello del 16 giugno 2016

Esercizio1
class Path implements Cloneable{
    private int n;
    private Path q;
        //AF(c) = 0 se c.n==0
        //AF(c) = k.q se k=c.n and q=AF(c.p) se c.n!=0
        //I(c) = n≥0 and n!=0 sse q!=null
    public Path(){n=0;
    }
    public Path add(int x){
        if(x<=0)return this;
        Path res = new Path();
        if(n==0){res.q=this; res.n=x;
            return res;
        }
        res.n=n; res.q=q.add(x);
        return res;
    }
    public Path clone() throw CloneNotSupportedException{
        return ((Path) super.clone());
    }
    public boolean equals(Object p){
        if (p==null) return false;
        try{Path myp = (Path) p;
            return ((n==myp.n) && q.equals(myp.q));
        }
        catch(Exception e){};
        return false;
    }

Esercizio2
exception InvalidOperationException;;
module type PATH =
sig type path
    val zero: unit -> path
    val add: int -> path -> path
    val addP: path -> path -> path
    val isEmpty: path -> bool
    val next: path -> int
    val rest: path -> path
end;;
module Path =
(struct
    type path = int list
        (* AF(c) = 0 se c=[]
           AF(c) = n.p se c=n::r && p=AF(r)
```

```
           I(c) = (c=[])||((hd c)>0 && I(tl c))
        *)
    let zero() = []
    let add n p =
        if(n<=0)p
        else match p with
                | [] -> n::[]
                | m::q -> m::(add n q)
    let addP p q = p @ q
    let isEmpty p = (p=[])
    let next p = match p with
            | n::q -> n
            | _ -> raise (InvalidOperationException)
    let rest p = match p with
            | n::q -> q
            | _ -> raise (InvalidOperationException)
end:PATH)

Esercizio3 - soluzione breve
class ParseTreeADTE<A,B extends A> extends ParseTreeADT<A,B>{
        //AF(c) = AF(c.super)
        //I(c) = true
    public ParseTreeADTE(){
        super();
    }
    public ParseTreeADTE(A v){
        super(v);
    }
    public ParseTreeADTE(B v, LinkedList<ParseTreeADT<A,B>> sons){
        super(v,sons);
    }
    //isEmpty, isLeaf, isRoot, getLabel, getSons, set, additionals ereditati
    public int size(){
        int ret = 0;
        for(ParseTreeAPI<A,B> u: elements()){
            if (u.isRoot()) ret++;
        }
        return ret;
    }

```