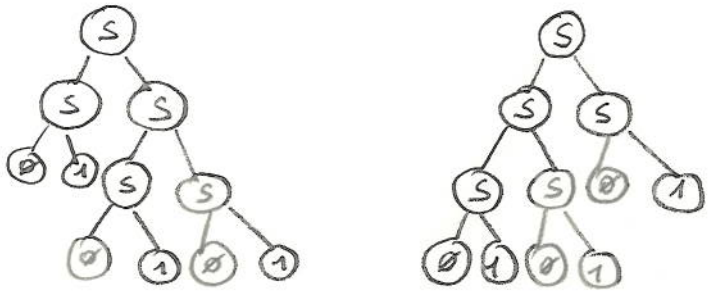
1) La stringa 010101 ha due alberi di derivazione



$S \to A \mid AS$

$A \to 01 \mid 0S1$

2)

```
let rec rt2lst rt = match rt with
      Void -> []
    | Node(x, nt) -> x :: rt2lst nt ;;
```

3)

```
int index (int a[], int dim, int x)
{ int i = 0; int trovato = 0;
  int sum = 0;
  while (i < dim && !trovato)
    if (sum < x && x < sum + a[i]) trovato = 1;
      else { sum = sum + a[i];
             i = i + 1;
           }
  if (trovato) return i;
     else      return -1;
}
```

4) La funzione sulle liste vuote è indefinita

```
let max_occ l =
        let f (v,m) (v1, m1) =
            if m > m1 then (v, m)
                      else (v1, m1)
        in
            match l with
            (x, m) :: xs ->
                let (v,m) = foldr f (x,m) xs
                in v ;;
```

5)

```
int check (int a[], int dim)
{ int count = 0; int appartiene = 1;
  int i = 0;
  while (i < dim && count >= 0 && appartiene)
   if (a[i] != 0 && a[i] != 1) appartiene = 0;
     else{if (a[i]==1) count = count + 1;
          else          count = count - 1;
          i = i + 1
          }
  return (i == dim) && (count == 0);
}
```

6)

```
Void totop ( ListaDiElementi *l)
{ if (*l != NULL)
    if (*l->next != NULL)
      if (*l -> info <= *l -> next -> info)
        { ListaDiElementi prec = *l;
          ListaDiElementi corr = prec -> next;
          int trovato = 0;
          while ( corr -> next != NULL && ! trovato)
            if (corr -> info > corr -> next -> info)
                trovato = 1
            else { prec = corr;
                   corr = corr -> next;}
          if (trovato)
            { prec -> next = corr -> next;
              corr -> next = *l;
              *l = corr;
            }
        }
}
```