Es.1.

$G' = \langle \{a,b,c\}, V \cup \{S'\}, S', P \cup P' \rangle$

$P' = \{ S' \to S'c \mid Scc \}$

Es.2.

```
let rec mindepth bt x = match bt with
        Void -> ∅
  | Node (y, lbt, rbt) = if x = y then 1 else
      let ml = mindepth lbt x
      and mr = mindepth rbt x
      in
        if (ml = ∅) & (mr = ∅) then ∅
          else if (ml = 0) then 1 + mr
            else if (mr = 0) then 1 + ml
              else 1 + min (ml, mr);;
```

dove min è l'ovvia funzione.

Es. 3.

```
int verify ( int a[], int dim)
{ int i=0; int trovato=0;
  while (i<dim && !trovato)
  { int trovatouguale = 0;
    int tuttidiversi = 1;
    int j=0;
    while (j<i && !trovatouguale)
      if (a[j]==a[i]) trovatouguale =1;
       else j=j+1;
    if (trovatouguale)
    { j = i+1;
      while ( j<dim && tuttidiversi)
        if (a[j] == a[i]) tuttidiversi = 0;
         else j=j+1;
    }
    if (trovatouguale && tuttidiversi) trovato=1;
     else i = i+1
  }
return trovato;
}
```

Es. 3. (seconda versione)

```
int verify (int a[], int dim)
{ int i = dim-1; int trovato = 0;
  while (i > 0 && !trovato)
  { int j = i-1;
    while (j >= 0 && !trovato)
      if (a[j]==a[i]) trovato = 1;
        else j = j-1;
    if (!trovato) i = i-1;
  }
  return trovato;
}
```

Es. 4.

```
void insmin (int a[], int b[], int min[], int dim)
{ int ia = 0; int ib = 0; int imin = 0;
  while (imin < dim)
  { if (a[ia] < b[ib])
      { min[imin] = a[ia];
        ia = ia + 1;
      }
    else
      { min[imin] = b[ib];
        ib = ib + 1;
      }
    imin = imin + 1;
}
```

Es. 5.

```
let mapif p f lis = let g x y =
                        if p x then f x :: y
                               else x :: y
                    in foldlr g [] lis ;;
```

oppure

```
let mapif p f lis = let g x =
                        if p x then f x
                               else x
                    in map g lis ;;
```

Es 6.

```
void cancel (listaDiElementi *l)
{ if (*l != NULL)
    if (*l->next != NULL)
    if (*l->info < *l->next->info)
    { listaDiElementi aus = *l;
      *l = *l->next;
      free (aus);
    }
  else
    { int trovato = 0;
      listaDiElementi prec = *l;
      listaDiElementi curr = *l->next;
      while (curr->next != NULL && !trovato)
       if (curr->info < curr->next->info)
       { prec->next = curr->next;
         free (curr);
         trovato = 1;
       }
       else
       { prec = prec->next;
         curr = curr->next;
       }
    }
}
```