Algoritmos e Complexidade
LEI/LCC (2º ano)

3ª Ficha Prática

Ano Lectivo de 2009/10

O objectivo desta ficha é a escrita de variantes e invariantes que permitam provar a correcção (total) de algoritmos que envolvam ciclos.

1. Para cada um dos programas seguintes determine um variante e um invariante que lhe permita (apenas) provar a terminação dos ciclos em causa. Determine ainda a pré-condição necessária a que o ciclo termine de facto.

   (a)
   ```
   WHILE (I < N) DO
   BEGIN I:=I+1; S:=S*2
   END
   ```

   (b)
   ```
   R:=X;
   Q:=0;
   WHILE (Y <= R) DO
   BEGIN R:=R-Y; Q:=Q+1
   END
   ```

   (c)
   ```
   RES := 0;
   WHILE (Y>0) DO
   BEGIN RES := RES + X;
         Y = Y-1
   END
   ```

   (d)
   ```
   RES := 0;
   WHILE (Y>0) DO
   BEGIN IF (Y % 2 != 0) THEN
         BEGIN Y := Y - 1;
               RES := RES + X
         END
         X := X*2;
         Y := Y/2
   END
   ```

   (e)
   ```
   Min = A[1][1];
   I := 1;
   WHILE (I<=N) DO
   BEGIN J := 1;
         WHILE (J<=N) DO
         BEGIN IF (Min > A[I][J]) THEN Min := A[I][J]
               J := J + 1
         END;
         I := I+1
   END
   ```

   (f)
   ```
   Min = A[1][1];
   I := 1; J:= 2
   WHILE (I<=N) DO
   BEGIN IF (Min > A[I][J]) THEN Min := A[I][J]
         J := J + 1;
         IF (J > N) THEN
         BEGIN J := 1; I:= I+1
         END
   END
   ```

2. Determine as condições de verificação necessárias para provar a correcção total dos seguintes algoritmos anotados.

(a)
```
// x >= 0 && y > 0
r := x;
// x >= 0 && y > 0  && r == x
WHILE (r>=y) DO
   // r >= 0 && (\exists q >= 0 : q * y + r = x) ; r
   r := r - y;
// 0 <= r < y && (\exists q >= 0 : q * y + r = x)
```

(b)
```
// n >= 0
k = 1; i=0;
// n > 0 && k = 1
WHILE (i < n) DO
BEGIN // i <= n 0 && k = i! ; n-i
      i:=i+1; k:=k*i
END
// k = n!
```

(c)
```
// n > 0
i = n-1; k = 0;
// n > 0 && i = n-1 && k = 0
WHILE (i > 0) DO
BEGIN // i >= 0 && k < n ; i
      IF (a[i] < a[i-1]) THEN
      BEGIN t:=a[i]; a[i]:=a[i-1];a[i-1]:=t;
            k=k+1
      END;
      i:=i-1;
END
// k < n
```