Lista de Exercícios #12

Geração de TAC, Checagem de Tipos, Expressões Booleanas, Controle de Fluxo

1. Determine os endereços relativos para os identificadores seguintes considerando o esquema de tradução abaixo:

   - `float x;`
   - `record { float x; float y; } p;`
   - `record { int tag; float x; float y; } q;`

P	→	M D
M	→	ϵ `{ offset = 0; }`
D	→	T id; `{ top.put(id.lexeme, T.tipo, offset); offset += T.width; }` D_1
D	→	ϵ
T	→	**record** '{' X D '}' Y
X	→	ϵ `{ Env.push(top); top = new Env(); Stack.push(offset); offset = 0; }`
Y	→	ϵ `{ T.tipo = record(top); T.width = offset; top = Env.pop(); offset = Stack.pop(); }`

2. Utilize o esquema de tradução abaixo para construir diferentes tipos válidos:

T	→	B `{ t = B.type; w = B.width; }` C
B	→	**int** `{ B.type = integer; B.width = 4; }`
B	→	**float** `{ B.type = float; B.width = 8; }`
C	→	ϵ `{ C.type = t; C.width = w; }`
C	→	**[num]** C_1 `{ array(num.value,` C_1`.type); C.width = num.value *` C_1`.width; }`

3. O que é avaliação em curto-circuito? Quais suas vantagens e desvantagens?

4. Qual a diferença entre avaliação numérica e por controle considerando expressões booleanas?

5. Considerando o esquema de tradução para avaliação numérica de expressões booleanas abaixo, gere código intermediário as seguintes expressões lógicas:

   - a < b and not c > d
   - a < b or c < d and e < f
   - not a < b or not c > d and x < q
   - not (a < b or not c > d) and x < q

```
E → E1 or E2      { E.nome = temp();
                    gera(E.nome = E1.nome or E2.nome }
E → E1 and E2     { E.nome = temp();
                    gera(E.nome = E1.nome and E2.nome }
E → not E1        { E.nome = temp();
                    gera(E.nome = not E1.nome }
E → (E1)          { E.nome = E1.nome }
E → E1 op E2      { E.nome = temp();
                    gera(if E1.nome op.simb E2.nome goto proxq+3);
                    gera(E.nome = 0);
                    gera(goto proxq+2);
                    gera(E.nome = 1); }
E → true          { E.nome = temp();
                    gera(E.nome = 1); }
E → false         { E.nome = temp();
                    gera(E.nome = 0); }
```

6. Avalie as expressões do exercício 5. considerando o esquema para avaliação por fluxo de controle abaixo.

```
B  →  { B1.t=B.t; B1.f=rot(); } B1 or { B2.t=B.t; B2.f=B.f; } B2
      { B.code=B1.code || label(B1.f) || B2.code }
B  →  { B1.t=rot(); B1.f=B.f; } B1 and { B2.t=B.t; B2.f=B.f; } B2
      { B.code=B1.code || label(B1.t) || B2.code }
B  →  not { B1.t=B.f; B1.f=B.t; } B1 { B.code=B1.code; }
B  →  (B1) { B.code=B1.code; B.t=B1.t; B.f=B1.f; }
B  →  true { B.code=gera(goto B.t); }
B  →  false { B.code=gera(goto B.f); }
B  →  E1 relop E2 { B.code=E1.code || E2.code ||
      gera(if E1.local relop.lexval E2.local goto B.t) ||
      gera(goto B.f); }
```

Considerando o esquema de tradução abaixo:

```
S  →  attr { S.code=gera(attr.lexval) || gera(goto S.next) }
S  →  if { B.t=rot(); B.f=S.next; }
      (B) { S1.next=S.next; }
      S1 { S.code=B.code || gera(B.t:) || S1.code }
S  →  if { B.t=rot(); B.f=rot(); }
      (B) { S1.next=S.next; }
      S1 else { S2.next=S.next; }
      S2 { S.code=B.code || gera(B.t:) || S1.code ||
           gera(B.f:); || S2.code }
S  →  while { B.f=S.next; B.t=rot(); }
      (B) { S.begin=rot(); S1.next=S.begin; }
      S1 { S.code=gera(S.begin:) || B.code ||
           gera(B.t:) || S1.code || gera(goto S.begin) }
B  →  { B1.t=B.t; B1.f=rot(); } B1 or { B2.t=B.t; B2.f=B.f; } B2
      { B.code=B1.code || label(B1.f) || B2.code }
B  →  { B1.t=rot(); B1.f=B.f; } B1 and { B2.t=B.t; B2.f=B.f; } B2
      { B.code=B1.code || label(B1.t) || B2.code }
B  →  not { B1.t=B.f; B1.f=B.t; } B1 { B.code=B1.code; }
B  →  (B1) { B.code=B1.code; B.t=B1.t; B.f=B1.f; }
B  →  true { B.code=gera(goto B.t); }
B  →  false { B.code=gera(goto B.f); }
B  →  E1 relop E2 { B.code=E1.code || E2.code ||
      gera(if E1.local relop.lexval E2.local goto B.t) ||
      gera(goto B.f); }
```

Gere o código TAC para os trechos de código seguintes:

1.
```
if (not (a < b or not c > d) and x < q) {
   x = z;
}
```

2.
```
while (a < b && e != f) {
  if (c < d){
    x = y + z;
  }else{
    x = x - z;
  }
}
```

3.
```
if (x > a){
  x = a;
}else{
  x = q;
}
```

Altere o esquema de tradução para fluxo de controle acima e adicione regras de tradução para construções do tipo:

1.
```
if (not (a < b or not c > d) and x < q) {
  x = z;
}else if (x > b){
  x = k;
}else if (a > e){
  x = q;
}
```

2.
```
if (x > a){
  x = a;
}else if (x > b){
  x = k;
}else{
  x = q;
}
```

3.
```
for (i = x; x < a; s = a){
  if (x > a){
    x = a;
  }
}
```

4.
```
switch (a){
  case d: x = c;
  case b: x = b;
  default: x = c;
}
```

Traduza para TAC o código acima utilizando o esquema de tradução modificado.