```
S → a+G     {   S.addr = newTemp()
                S.code = G.code || gen(S.addr '=' a.lexeme '+' G.addr)
            }
G → A*H     {   G.addr = newTemp()
                G.code = A.code || H.code || gen(G.addr '=' A.addr '*' H.addr)
            }
H → C-F     {   H.addr = newTemp()
                H.code = C.code || F.code || gen(H.addr '=' C.addr '-' F.addr)
            }
A → B+D     {   A.addr = newTemp()
                A.code = B.code || D.code || gen(A.addr '=' B.addr '+' D.addr)
            }
B → F*b     {   B.addr = newTemp()
                B.code = F.code || gen(B.addr '=' F.addr '*' b.lexeme)
            }
C → C/E     {   C.addr = newTemp()
                C.code = C1.code || E.code || gen(C.addr '=' C1.addr '*' E.addr)
            }
C -> l      {   C.code = ''
                C.addr = l.lexeme
            }
D → D * F   {   D.addr = newTemp()
                D.code = D1.code || F.code || gen(D.addr '=' D1.addr '*' F.addr)
            }
C -> h      {   C.code = ''
                C.addr = h.lexeme
            }
E → -A      {   E.addr = newTemp()
                E.code = A.code || gen(E.addr '=' 'minus' A.addr)
            }
F → -k      {   F.addr = newTemp()
                F.code = gen(F.addr '=' 'minus' k.lexeme)
            }
```