Domanda 1. (8 punti)

Usando le tabelle allegate, costruisci il codice a tre indirizzi del seguente frammento di programma:

```
z = 5
while (x && z > 2)
    z = z - 1
```

Spiega i vari passi del procedimento a partire dalla costruzione del parse-tree.

Domanda 2. (8 punti)

Data la grammatica G:

$$\begin{aligned} S &\rightarrow EF \mid FC \\ E &\rightarrow FE \mid a \\ F &\rightarrow CC \mid b \\ C &\rightarrow EF \mid a \end{aligned}$$

(a) dimostrare che G non è LL(1);

(b) costruire la tabella di parsing SLR;

(c) individuare eventuali conflitti shift-reduce;

(d) simulare il parser SLR sulla stringa bab.

Domanda 3. (8 punti)

Considera la grammatica G:

$$\begin{aligned} A &\rightarrow -A \mid (A) \mid CB \\ B &\rightarrow -A \mid \varepsilon \\ C &\rightarrow a\ D \\ D &\rightarrow (A) \mid \varepsilon \end{aligned}$$

(a) Dimostrare, applicando la definizione, che G è LL(1);

(b) simulare il parser LL(1) su input `a(-a)`.

Domanda 4. **(6 punti)**

Scegli una delle espressioni del linguaggio Fasto (definite in SML nel file `Fasto.sml` del progetto sviluppato in laboratorio) tra quelle riportate in Figura 1 e definisci per quella espressione

- l'Interprete `evalExp`;
- il Type Checker.

```
datatype Exp =
...

| Iota of Exp * pos                                          (* iota(n) *)
| Map of FunArg * Exp * T.TypeAnnot * T.TypeAnnot * pos   (* map(f, arr) *)
| Reduce of FunArg * Exp * Exp * T.TypeAnnot * pos  (* reduce(f, 0, arr) *)
```

Figure 1: Frammento di `Fasto.sml`