Domanda 1. (8 punti)

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

```
z = 5
if ( x )
    z = z - 1
 else
    z = z + 1
```

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

Domanda 2. (8 punti)

Data la grammatica G:

$$\begin{aligned} S &\rightarrow AB \mid BC \\ A &\rightarrow a \\ B &\rightarrow b \\ C &\rightarrow AB \mid a \end{aligned}$$

(a) stabilire se G è LL(1) applicando la definizione;

(b) costruire la tabella di parsing SLR;

(c) individuare eventuali conflitti shift-reduce;

(d) analizzare la stringa *abab* usando il parser opportuno in base alle proprietà della grammatica.

Domanda 3. (8 punti)

Considera la grammatica G:

$$\begin{aligned} S &\rightarrow rAB \\ A &\rightarrow aA \mid e \\ B &\rightarrow bB \mid e \end{aligned}$$

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

(b) Simulare il parser LL(1) su input `raeb`.

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`