Problema 1

Il problema richiede che venga individuata la differenza (positiva) tra il valore che i due ladri ottengono dividendosi il bottino.
Nella funzione main viene effettuata la lettura dei dati da considerare attraverso il terminale e viene fatto per ogni caso di test.
Per ogni caso di test viene anche invocata la funzione **MaxValue** che si occupa di individuare l'output richiesto. Nello specifico, la funzione prende in input l'array **arr[]**, due somme parziali **sum_1** e **sum_2**, e l'indice corrente **i** dell'elemento considerato nell'array.
La funzione utilizza la tecnica della programmazione dinamica per trovare la minima differenza tra **sum_1** e **sum_2**, tentando in considerazione tutte le possibili combinazioni di distribuzione delle monete nei due gruppi attraverso una ricorsione. Di particolare importanza è l'array **memo[].** La memoization ci evita di ricalcolare le soluzioni per sottoproblemi già risolti, salvando i risultati intermedi e utilizzandoli direttamente quando necessario.

Di seguito sono riportati 3 casi di test alternativi a quelli forniti:

Sample Input

```
3
1
22
4
2 3 5 6
3
7 2 4
```

Sample Output

```
22
0
1
```

Di seguito viene riportata l'analisi di complessità:

Complessità nel caso peggiore: O(nMoney)

Dove nMoney indica il numero di monete rubate. In particolare ogni sottoproblema ha una complessità di O(1) ma in base al numero di sottoproblemi diventa la complessità sopra riportata. Inoltre la complessità risulta tale nel caso di un singolo

caso di test. In presenza invece di ulteriori casi di test, la complessità assume la seguente forma: O(nTest*nMoney).

Elaborato svolto da:
Marco Dell'Isola M63001637
Raffaele Cuzzaniti M63001614