Problem 1

a) Given a hash function

$$h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod m,$$

where $h_1(k) = k \bmod 5$, $h_2(k) = (7 \cdot k) \bmod 2$ and a hash table of size $m = 5$. The order of open addressing of the sequence $< 3, 10, 2, 4 >$ is as follows:

$$h(3, i) = h_1(3) = (3 \bmod 5) \bmod 5 = 3$$

$$< Nil, Nil, Nil, 3, Nil >$$

$$h(10, i) = h_1(10) + i \cdot h_2(10) = (10 \bmod 5) \bmod 5 = 0$$

$$< 10, Nil, Nil, 3, Nil >$$

$$h(2, i) = h_1(2) + i \cdot h_2(2) = (2 \bmod 5) \bmod 5 = 2$$

$$< 10, Nil, 2, 3, Nil >$$

$$h(4, i) = h_1(4) + i \cdot h_2(4) = (4 \bmod 5) \bmod 5 = 4$$

$$< 10, Nil, 2, 3, 4 >$$

As the first hash function successfully mapped the key to unique positions in the hash function, there were no colissions and the second hash function was never called to resolve a conflict. The final hash table looks like:

$$< 10, Nil, 2, 3, 4 >$$

b) Please find the implementation in the "HashTable" folder.

Problem 2

a) Given a greedy algorithm that selects activies with the shortest local run time in order to maximise the number of activities executed, assume an activity list:

$$\{(3,6),(1,4),(1,2),(6,8),(0,2)\}$$

The process with the shortest local run time is $(1,2)$, the next compatible process with the shortest run time is $(6,8)$. The total number activities that are run with this algorithm is 2. However, we can see that more than 2 activities can be performed. The activities:

$$\{(0,2),(3,6),(6,8)\}$$

are compatible. Therefore, a this greedy algorithm does not always produce a global optimum solution.

b) The pseudo code below implements a greedy algorithm that selects activities with the latest starting time.

```
SelectActivity (A)
i = 0
//Sort run times from largest to smallest start time
for i = 1 to n − 1{
    for j = 0 to n − 2{
        if (A[j].start < A[j+1].start)
        swap (A[j], A[j+1])
    }
}

//Select the first activity
Select A[0]

//Find next activity
i = 0
for j = 1 to n − 1{
    if (A[j].finish ≤ A[i].start)
        Select A[j]
    i = j
}
```