York University CSE 3101 October 29, 2008

Homework Assignment #7
Due: November 7, 3:30 p.m.

1. Tiidrek is buying candy to give to children on Hallowe'en. He has n dollars to spend. In the store there are k types of candy packs. The ith type of pack contains a_i candies and costs c_i dollars (for $1 \leq i \leq k$). The store has an unlimited supply of each type of pack. The store sells only complete packs. (You cannot buy half a pack, for example). Tiidrek will give one piece of candy to each child that comes to his door. (Different children might get different types of candy.) He wants to maximize the number of children he can give candies to.

(a) Tiidrek comes up with the following strategy. He defines the "per-candy" price of each candy type to be $p_i = \frac{c_i}{a_i}$. By chance, it turns out that they are all distinct. He sorts the types so that $p_1 < p_2 < p_3 < \cdots < p_k$. Then he runs the following algorithm:

```
for $i = 1..k$
    buy $\left\lfloor \frac{n}{c_i} \right\rfloor$ packs of type $i$
    $n = n - c_i \cdot \left\lfloor \frac{n}{c_i} \right\rfloor$ // update amount of money left
end for
```

Show that Tiidrek's solution does not always yield the optimal result, even when $k = 2$.

(b) Explain how to use dynamic programming to efficiently compute the maximum number of candies Tiidrek can buy. You should define what the entries of your matrix represent, and then explain how to fill the entries in. (Explain briefly why your algorithm fills in the array correctly, but do not give a formal proof.)

2. Tiidrek's brother, Hindrek, is playing a game. Two shuffled decks of cards are placed side by side, with the cards face up. At any time, if the top card of the left deck is the same as the top card of the right deck, Hindrek can take both top cards and put them in his pocket. If the two top cards are different, Hindrek can remove either the top card of the left deck or the top card the right deck and discard it. He continues until all cards are either discarded or in his pocket. A king, queen or jack card is worth 10 points. An ace is worth 11 points. A 2 is worth 2 points, a 3 is worth 3 points, a 4 is worth 4 points, and so on. Hindrek's goal is to maximize the total number of points for all the cards that end up in his pocket.

Hindrek is allowed to peek before playing: he can look through the entire decks before beginning (but he cannot change the order of the cards in the decks). Let $L[1..52]$ and $R[1..52]$ to be the cards in the left and right decks, respectively, in order from the top of the deck to the bottom.

Explain how dynamic programming can be used to efficiently compute Hindrek's best possible score, given the arrays L and R. You should define what the entries of your matrix represent, and then explain how to fill the entries in. (Explain briefly why your algorithm fills in the array correctly, but do not give a formal proof.)