Department of Computer Engineering, METU

CENG 140

Fall 2021-2022

THE 1

Görkem Özer
gorkem@ceng.metu.edu.tr

Due date: 20 December 2021, Monday, 23:59

1 Part 1 - Card game (50 points)

Gürbüz is a talented computer engineering student who loves to develop games. He invented a card game, in which two players receive 3 hidden cards from cards at the beginning. There are 100 cards and each card has a number between **1-100** written on it. A player does not know what cards the other opponent received. The game starts after players receive their cards.

There will be 3 rounds. On each round, the players will pick a card among 3 cards that they received in the beginning. If a player has picked a card with value greater than the opponent's choice, then the player wins the round. Whoever wins 2 rounds out of 3 wins the game.

Nazlı is also another computer engineering student and heard about the game and found a serious bug. Now, she has the chance to interfere before she receives the last card from the game. She can see all 5 cards that the game assigned until now and she can make the game assign whichever card she wants as her last (third) card.

Nazlı wants to mock with Gürbüz. She wants to win every time or wants to withdraw if she is going to lose inevitably. How could she do this? Write an **ITERATIVE** solution for solving this problem. The input has 5 cards between 1-100, first 3 cards are Gürbüz's cards and last 2 cards are Nazlı's cards. **You need to find a card that makes Nazlı win the game no matter how badly she plays. If there are multiple cards that guarantee a win for the game, print the card with the minimum value. If there is no card that guarantees a win for the game, you need to print "WITHDRAW".**

Input Format:

```
<card1_gurbuz> <card2_gurbuz> <card3_gurbuz> <card1_nazli> <card2_nazli>
```

Output Format:

```
<result>
```

Input 1:

```
39 62 40 61 64
```

Output 1:

```
41
```

Input 2:

```
67 43 36 27 44
```

Output 2:

```
WITHDRAW
```

Input 3:

```
41 51 61 55 57
```

Output 3:

```
52
```

Input 4:

```
94 96 97 95 98
```

Output 4:

```
99
```

2 Part 2 - Strengthen the Army (with recursion, 50 points)

A kingdom's strength always came from its treasury, its army and its dedication to science and technology. Imagine a kingdom that a king decided to improve the capability of fighting of his army. Since the kingdom is under the threat of an enemy, the king orders this command to his generals:

"You shall find me the best way to strengthen the army. The army must be strong and ready for war at once!".

The counsellors provisioned that the available budget for this task is **N** coins. Here are the options to strengthen the army, each has different cost and different power boost:

- Enlisting new people to the army (option 1)
- Upgrading the armor of archers and infantry (option 2)
- Investing on cavalry, buying fast horses (option 3)
- Researching for better aiming mangonels for sieges (option 4)
- Upgrading the cannons against towers (option 5)
- Improving the equipment used by the infantry (option 6)

So, if you are asked to make choices, how would you implement a strategy for picking the options, which lead to maximum power gain within the given budget (i.e. amount of coins)? Your task in this part is to implement a **RECURSIVE** solution to this problem. Your recursive solution should find the cost for the subset of options that yields the **MAXIMUM** power gain. Note that, as you look for the maximum power gain, you should consider each subset of options (using a recursive approach). Note that, as you look for the **OPTIMAL** power gain, you should consider each subset of options (using a recursive approach). *Hint: You may like to watch/read the lecture for recursion on our course page (week 6).* You should implement **the1_pt2.c**. Iterative solutions **WILL NOT** be accepted as an answer and will get 0.

You must consider these:

- Each improvement has a *cost* and *power boost values*.
- These *cost* and *power boost values* are variable, they will be provided to you as inputs.
- First input will be the number of coins in the budget.
- Each of the remaining lines defines the *cost* and *power boost value* of each option above.
- **Input Format:**

  ```
  <number_of_coins>
  ```

- **Output Format:**

  ```
  <maximum power gain>
  ```

- **Examples:**

  Input:

```
4800        - budget
800 150     - <cost> <power_boost> for option1
1200 200    - <cost> <power_boost> for option2
1500 250    - <cost> <power_boost> for option3
3500 300    - <cost> <power_boost> for option4
4000 350    - <cost> <power_boost> for option5
1000 275    - <cost> <power_boost> for option6
```

Output:

```
875
```

Note that for the input above; from all combinations; the combination of enlisting people (800, 150), upgrading armor (1200, 200), investing on cavalry (1500, 250) and improving the equipment (1000, 275) is optimal. The army gains 875 power with the cost of 4500 coins (which is under 4800).

3 Regulations

- **Programming Language:** C
- **Libraries and Language Elements:**
  You should not use any library other than the ones given you in the code. You can use conditional clauses (switch/if/else if/else), loops (for/while), arrays. **You can NOT use any further elements beyond those.** You can define helper functions.
- **Compiling and running:**
  You should be able to compile your codes and run your program with following commands:

  ```
  >_ gcc the1_pt1.c -ansi -pedantic-errors -Wall -o the1_pt1
  >_ ./the1_pt1
  >_ gcc the1_pt2.c -ansi -pedantic-errors -Wall -o the1_pt2
  >_ ./the1_pt2
  ```

  If you are working with ineks or you are working on Ubuntu OS, you can feed your program with input files instead of typing inputs. This is a feature of Ubuntu OS and an equivalent of typing inputs. This way you will test your program faster:

  ```
  >_ ./the1_pt1 < inp1.txt
  >_ ./the1_pt1 < inp2.txt
  ```

- **Submission:**
  You should submit your the1_pt1.c and the1_pt2.c implementations. Late submission IS NOT allowed, it is not possible to extend the deadline and **please do not ask for any deadline extensions**.
- **Evaluation:**
  Your codes will be evaluated based on several inputs including, but not limited to the test cases given to you as example. If your program gives correct outputs for all cases, you will get 100 points.
- **Cheating:**
  We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations and will get 0. Sharing code between each other or using third party code is strictly forbidden. Even if you take a "part" of the code from somewhere/somebody else - this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar. So please do not think you can get away with by changing a code obtained from another source.