გამოცდის ფორმატი

***მონიშნეთ გამოცდის ფორმატი (მიუთითეთ √)**

დახურული წიგნი	
ღია წიგნი	√

***ღია წიგნის შემთხვევაში მონიშნეთ გამოცდაზე ნებადართული ელემენტები (მიუთითეთ √)**

სალექციო მასალები (პრეზენტაცია და სხვა)	
ელექტრონული წიგნები	√
წიგნები	
კონსპექტები	
ლექსიკონი	
კალკულატორი	
ლეპტოპი/პლანშეტი	

*** გამოცდის ჩატარების წესი იხილეთ ,,დესკტოპზე“ საქაღალდეში Exam materials**

საგამოცდო საკითხების ფორმა
ვარიანტი # 1

სკოლა/ საგანმანათლებლო პროგრამა	მათემატიკა და კომპიუტერული მეცნიერება	***სტუდენტის მიერ მიღებული ქულა***	
საგანი	პროგრამირების პარადიგმები		
ლექტორი	შ. ღვინეფაძე		
კურსი	II		
ჯგუფი			
გამოცდის ფორმა	ღია წიგნი		
გამოცდის ხანგრძლივობა	3 საათი		
მაქსიმალური ქულა	130		
სტუდენტის სახელი და გვარი:			

სახელი: ქულა:

შუალედური გამოცდა
პარადიგმებში
2018, 17 ნოემბერი 14:40 – 17:40

1 40 ქულა	2 90 ქულა	სულ

შეასრულეთ შემდეგი ინსტრუქციები, წინააღმდეგ შემთხვევაში შესაძლოა თქვენი ნაშრომი არ შეფასდეს.
1. ჩამოტვირთეთ paradigms-midterm2 ფოლდერი თქვენს დესკტოპზე. მასში უნდა იყოს 2 ფოლდერი blockchain და assembly თავისი შესაბამისი ფაილებით.
2. ცვლილებები შეიტანეთ დავალების პირობით მითითებულ ფაილებში.
3. დააარქივეთ paradigms-midterm2 ფოლდერი, არქივს სახელად დაარქვით თქვენი მეილის პრეფიქსი, მაგალითად gboch12.rar.
4. ვებ ბრაუზერში გახსენით მისამართი http://192.168.210.5 და ატვირთეთ არქივი.

Command prompt-ის გამოსაძახებლად ფოლდერში Shift ღილაკთან ერთად დააკლიკეთ მაუსის მარჯვენა ღილაკს და აირჩიეთ open command prompt here

ამოცანა 1. ასემბლერი (40 ქულა)

შექმენით .txt ფაილი და დაწერეთ ასემბლის კოდი Transfer ფუნქციისთვის. ცალკე გამოყავით თითოეული ხაზის შესაბამისი კოდი

```
typedef struct Transaction{
        char * from;
        char to[8];
        int amount;
} Transaction;

int hash(void *item);

int Transfer(struct Transaction trans, int * accounts)
{
        int index = hash(trans.from);
        if (accounts[index] > trans.amount){
                accounts[index] = accounts[index]-trans.amount;
                accounts[trans.to[0]] = accounts[(int)trans.to]+trans.amo
```

ასემბლის კოდი დაწერეთ asslembly ფოლდერში მდებარე transfer.txt ფაილში.

ამოცანა 2. ბლოქჩეინი (90 ქულა)

თქვენი ამოცანაა ბლოქჩეინ(blockchain) სტრუქტურის იმპლემენტაცია. ბლოქჩეინი არის მონაცემთა სტრუქტურა, რომლის საშუალებითაც შესაძლებელია მომხმარებლების ანგარიშებისა და გადარიცხვების შენახვა.

მომხმარებლის ანგარიშის იდენტიფიკაცია შესაძლებელია მისი სახელის მიხედვით, რაც ჩვენს შემთხვევაში არის უბრალო C სტრინგი. არსებობს ტრანზაქციის ცნება. ტრანზაქცია არის გადარიცხვა ერთი მომხმარებლის ანგარიშიდან, მეორე მომხმარებლის ანგარიშზე. ტრანზაქცია მოიცემა შემდეგი მონაცემთა სტრუქტურით:

```
typedef struct Transaction {
	char * from;
	char * to;
	int amount;
} Transaction;
```

from ანგარიშდან to ანგარიშზე amount ოდენობის თანხის გადარიცხვა. ტრანზაქციები ერთიანდება ბლოკში. ბლოკში ნებისმიერი რაოდენობის ტრანზაქცია შეიძლება იყოს. ბლოკისათვის შესაძლებელია ჰეშ ფუნქციის დათვლა, რომელიც აბრუნებს int-ს. ფუქცია მოცემულია შესაბამის ფაილში. ბლოკი ტრანზაქციების გარდა შეიცავს წინა ბლოკის ჰეშს. ბლოკი მოიცემა შემდეგი სტრუქტურით:

```
typedef struct Block {
	vector * transactions;
	int previousHash;
} Block;
```

ბლოქჩეინი ბლოკების ვექტორია. ბლოკები previousHash-ის საშუალებით გადაბმულია ერთმანეთზე და შეგვიძლია შევამოწმოთ, ვექტორის შემდეგ ბლოკში ჩაწერილი previousHash ემთხვევა თუ არა წინა ბლოკში ჩაწერილ ჰეშს. ანუ: vector[i].previousHash == hash(vector[i-1]).
თქვენი ამოცანაა ბლოქჩეინისთვის შემდეგი ფუნქციების იმპლემენტაცია:
1. addBlock - გადაეცემა ახლი ბლოკი და ამატებს მას ბლოქჩეინის ბოლოში.
2. isValid - ამოწმებს ბლოქჩეინი არის თუ არა ვალიდური ანუ ყოველი ბლოკის ჰეში ემთხვევა თუ არა მომდევნო ბლოკის previousHash-ს. ამ ფუნქციის იმპლემენტაციისთვის არ უნდა გამოიყენოთ ციკლი და რეკურსია. გამოიყენეთ map ფუქცია.
3. getMaxBalance - ფუნქციამ უნდა გააანალიზოს მთელი ბლოქჩეინი, ანუ ყველა ბლოკის ყველა ტრანზაქცია. დაადგინოს რომელ მომხმარებელს აქვს ყველაზე დიდი თანხა ანგარიშზე და დააბრუნოს ამ მომხმარებლის სახელი. getMaxBalance ფუნქცია უნდა შესრულდეს O(n) დროში, სადაც n ტრანზაქციების რაოდენობაა.
ჩათვალეთ, რომ თავიდან ყველას ანგარიშზე თანხა არის 0. როდესაც გადარიცხვა ხდება ერთი ანგარიშიდან მეორეზე პირველ ანგარიშს თანხა აკლდება მეორეს კი ემატება. ჩათვალეთ, რომ უარყოფითი თანხები არსებობს. ანუ თუკი ვინმეს 0 აქვს ანგარიშზე მას მაინც შეუძლია თანხის გადარიცხვა და უარყოფითი ოდენობის ფული ექნება.
vector -ის და hashset-ის იმპლემენტაციები მოცემული გაქვთ და შეგიძლიათ გამოიყენოთ.