***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*** | 2 საათი | | |
| ***მაქსიმალური ქულა*** | 120 | | |
| ***სტუდენტის სახელი და გვარი:*** | | | |

სახელი: ქულა:

ფინალური გამოცდა

პარადიგმებში

2019, 22 იანვარი 10:00 – 12:00

შეასრულეთ შემდეგი ინსტრუქციები, წინააღმდეგ შემთხვევაში შესაძლოა თქვენი ნაშრომი არ შეფასდეს.  
1. ჩამოტვირთეთ paradigms-exam1 ფოლდერი თქვენს დესკტოპზე. მასში უნდა იყოს 2 ფოლდერი scheme და threads.

2. ცვლილებები შეიტანეთ დავალების პირობით მითითებულ ფაილებში.

3. დააარქივეთ paradigms-exam1 ფოლდერი, არქივს სახელად დაარქვით გამოცდის ნომერი და თქვენი მეილის პრეფიქსი, მაგალითად 1gboch12.rar.

4. ვებ ბრაუზერში გახსენით მისამართი [http://192.168.210.5](https://l.messenger.com/l.php?u=http%3A%2F%2F192.168.210.5%2F&h=ATNdNl_tUgM3RyoXqKtzPKRP12hygwyVI0pyQH9Q8ss9LJWHxdfFaufD0sZODW8AowpzwdCIygWTmkdTwBIdteLRo_hfKInsffPkNS6u9h_To4X4mdIiBDFsifj54UipvN3rxqw7KRPPZsVNx9Y) და ატვირთეთ არქივი.

5. არქივი დატოვეთ დესკტოპზე.

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

**ამოცანა 1. სქემა (50 ქულა)**

თქვენი ამოცანაა დაწეროთ ფუნქცია sum-set რომელაც გადაეცემა მთელი რიცხვების ლისტი, მთელი რიცხვი k და მთელი რიცხვი s და რომელიც აბრუნებს ლისტის ყველა k სიგრძის ქვესიმრავლეს, რომელთა ჯამიც s-ის ტოლია. უნდა დააბრუნოთ ლისტების ლისტი, სადაც თითოეული ლისტი უნიკალური ქვესიმრავლე იქნება.

მაგალითად

(sum-set ‘(1 2 3 4) 2 5) -> ‘((1 4) (2 3))

(sum-set ‘(2 2 1 5 6) 3 9) -> ‘((2 2 5) (2 1 6))

(sum-set ‘(1 1 1 1 1 1) 4 5) -> ‘()

(sum-set ‘(1 1 1 1 1 1) 4 4) -> ‘((1 1 1 1))

(sum-set ‘(1 2 3 4 5 6 7 8 9 10) 2 15) -> ‘((5 10) (6 9) (7 8))

დაბრუნებულ ლისტების ლისტში ელემენტების მიმდევრობას მნიშვნელობა არ აქვს. ჩათვალეთ, რომ გადაცემულ ლისტში ელემენტების რაოდენობა ნაკლებია 31-ზე.

**ამოცანა 2. ნაკადები (70 ქულა)**

დაწერეთ Sync სტრუქტურის რეალიზაცია, რომლის ფუქნციების საშუალებით შესაძლებელია ნაკადის(ების) დაბლოკვა/განბლოკვა.

სტრუქტურას აქვს ოთხი ფუნქცია:

Init(Sync \* s);

Wait(Sync \* s, int id);

Notify(Sync \* s, int id);

NotifyAll(Sync \* s);

Init(Sync \* s) - ფუნქციით ხდება სტრუტურის ინიციალიზაცია. რა სახის იქნება სტრუქტურა, თქვენ უნდა მოიფიქროთ.

Wait(Sync \* s, int id) - ფუნქცია ბლოკავს ნაკადს რომელშიც მოხდა მისი გამოძახება(და რომლის უნიკალური ნომერიც არის id). ჩათვალეთ რომ ყველა ნაკადს აქვს უნიკალური ნომერი. ნაკადი დაბლოკილი რჩება მანამ სანამ არ მოხდება Notify ან NotifyAll ფუნქციის გამოძახება.

Notify(Sync \* s, int id) - ფუნქცია განბლოკავს გადაცემული id ის მქონე ნაკადს, თუ იგი დაბლოკილი იყო იგივე s სტრუქტურით, ანუ Wait(s, id) გამოძახების საშუალებით. სხვა შემთხვევაში, Notify არაფერს არ ცვლის.

NotifyAll(Sync \* s) - ფუნქცია განბლოკავს ყველა ნაკადს რომელიც დაბლოკილია s სტრუქტურის მეშვეობით.

პირობის გასააზრებლად განვიხილოთ მაგალითი. დავუშვათ გვაქვს სამი ნაკადი, რომელთა id-ებია 1, 2 და 3. ასევე გვაქვს ერთი ცალი Sync \* s სტრუქტურა, რომელიც უკვე ინიციალიზირებულია.

პირველი ნაკადის დაბლოკვისთვის, ამავე ნაკადში უნდა მოხდეს Wait ფუნქციის გამოძახება და არგუმენტებად გადაეცეს s და 1, ანუ Wait(s, 1);

ანალოგიურად მეორე ნაკადის დაბლოკვისთვის, მეორე ნაკადში უნდა მოხდეს Wait(s, 2) ფუნქციის გამოძახება.

მესამე ნაკადიდან პირველი ნაკადის განბლოკვა შესაძლებელია, მესამე ნაკადში Notify(s, 1) გამოძახებით. ხოლო თუ გვინდა მესამედან ორივე დაბლოკილი ნაკადი გავაღვიძოთ მაშინ NotifyAll(s) უნდა გამოვიძახოთ.

თუ მესამე ნაკადიდან Notify ის ან NotifyAll ს გადავცემდით განსხვავებულ Sync სტრუქტურას, მაგალითად s2 ს, მაშინ პირველი ორი ნაკადი ისევ დაბლოკილი დარჩებოდა, რადგან მათი დაბლოკვა s ის გამოყენებით მოხდა.

ყურადღება მიაქციეთ რამდენიმე დეტალს:

1. Wait ფუნქციას id პარამეტრად, ყოველთვის გადაეცემა იმ ნაკადის იდენტიფიკატორი, რომელშიც ხდება Wait ფუნქციის გამოძახება.

2. Notify(s, id) და NotifyAll(s) ფუნქციებით ხდება მხოლოდ იმ დაბლოკილი ნაკადების გაღვიძება, რომელთა დაბლოკვაც მოხდა იგივე s სტრუქტურის მეშვეობით.