***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*** | 2.5 საათი | | |
| ***მაქსიმალური ქულა*** | 100 | | |
| ***სტუდენტის სახელი და გვარი:*** | | | |

სახელი: ქულა:

შუალედური გამოცდა

პარადიგმებში

2017, 19 დეკემბერი 14:40 – 17:10

|  |  |  |
| --- | --- | --- |
| 1  50 ქულა | 2  50 ქულა | სულ |
|  |  |  |

შეასრულეთ შემდეგი ინსტრუქციები, წინააღმდეგ შემთხვევაში შესაძლოა თქვენი ნაშრომი არ შეფასდეს.  
1. ჩამოტვირთეთ paradigms-midterm3 ფოლდერი თქვენს დესკტოპზე. მასში უნდა იყოს 2 ფოლდერი odd-divisors და operators თავისი შესაბამისი ფაილებით.

2. ცვლილებები შეიტანეთ დავალების პირობით მითითებულ ფაილებში.

3. დააარქივეთ paradigms-midterm3 ფოლდერი, არქივს სახელად დაარქვით თქვენი მეილის პრეფიქსი, მაგალითად gboch12.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.

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

სქემის ინტერპრეტატორის გასაშვებად შეგიძლიათ დააკლიკოთ დესკტოპზე არსებულ kawa-ს. თქვენი ამოცანის დასატესტად ინტერპრეტატორში ჩააკოპირეთ ფუნქცია და ისე შეამოწმეთ.

**ამოცანა 1. კენტი გამყოფები (50 ქულა)**

თქვენი ამოცანაა დაწეროთ ფუნქცია odd-divisors, რომელსაც გადაეცემა მთელი რიცხვი n და რომელიც აბრუნებს ამ რიცხვის კენტი გამოყოფების ჯამს. მაგალითად რიცხვი 24-ის კენტი გამყოფებია: 1 და 3, შესაბამისად კენტი გამყოფების ჯამი იქნება 4. ხოლო რიცხვის 17 კენტი გამყოფებია 1 და 17, შესაბამისად კენტი გამყოფების ჯამი იქნება 18.

ფუნქცია დაწერეთ პროგრამული ენა სქემის საშუალებით. ფუნქციის დაწერის დროს გამოიყენეთ მხოლოდ ფუნქციონალური პარადიგმა.

შეიძლება დაგჭირდეთ mod ფუნქციის გამოყენება, რომელიც რიცხვის მეორე რიცხვზე გაყოფის შედეგად მიღებულ ნაშთს აბრუნებს.

(define (odd-divisors n)

ტესტები:

(odd-divisors 24) => 4

(odd-divisors 17) => 18

(odd-divisors 15) => 24

**ამოცანა 2. ოპერატორები (50 ქულა)**

თქვენი ამოცანაა დაწეროთ საბანკო დღის სიმულაცია. სიმულაციაში მონაწილეობენ მომხმარებლები და ოპერატორები,მათი რაოდენობა მოცემულია NUM\_CUSTOMERS, NUM\_OPERATORS კონსტანტებით. მომხმარებლების ქცევა იდენტურია და მოცემულია Customer ფუნქციის საშუალებით. ოპერატორების ქცევაც იდენტურია და მოცემულია Operator ფუქნციის საშუალებით.

მომხმარებელი პირველ რიგში იღებს რიგის ნომერს და ელოდება სანამ რომელიმე ოპერატორი გამოიძახებს მას, შემდეგ ავსებს სესხის ფორმას და ელოდება სანამ ოპერატორი არ გადაამოწმებს მონაცემებს და ეტყვის პასუხს სესხის დამტკიცებაზე, ამის შემდეგ მომხმარებელი მიდის სახლში.ოპერატორები ელოდებიან მომხმარებლის გამოჩენას, როგორც კი მომხამრებელი ჩადგება რიგში, რომელიმე თავისუფალი ოპერატორი გამოიძახებს მას და ელოდება სანამ იგი შეავსებს აპლიკაციის ფორმას. შემდეგ ოპერატორი ამოწმებს მონაცემებს, ეუბნება დაადებით ან უარყოფით პასუხს მომხამრებელს და ელოდება მომდევნო მომხმარებელს.

ოპერატორები ემსახურებიან მომხმარებლებს ზუსტად იმ მიმდევრობით რა მიმდევრობითაც ჩადგნენ რიგში. როცა რამდენიმე ოპერატორი ელოდება მომხმარებელს, თუ მხოლოდ ერთი მომხმარებელი ჩადგება რიგში, მას თავისთან გამოიძახებს მომლოდინე ოპერატორებიდან ერთერთი(სულ ერთია რომელი), ხოლო დანარჩენები გააგრძელებენ ლოდინს.

threads.c ფაილში მოცემული გაქვთ საწყისი კოდი სადაც main ფუნქციიდან NUM\_CUSTOMERS ნაკადში ხდება Customer ფუქცნიის გამოძახება, ხოლო NUM\_OPERATORS ნაკადში Operator ფუნქციის გამოძახება.

ჩათვალეთ რომ fillApplication და checkApplication ფუნქციები უკვე არსებობს და გამართულად მუშაობს. შეგიძლიათ შემოიტანოთ თქვენთვის საჭირო გლობალური ცვლადები. შეფასებისას ყურადღება მიექცევა სიმულაციის ეფექტურობას, ნაკადების ლოდინი არ უნდა ხდებოდეს "Busy waiting"-ის საშუალებით.

#define NUM\_CUSTOMERS 20

#define NUM\_OPERATORS 4

int customersLeft = NUM\_CUSTOMERS;

void Customer() {

//TODO: რიგში ჩადგომა და ლოდინი სანამ რომელიმე ოპერატორი გამოიძახებს

fillApplication();

//TODO: ელოდება სანამ ოპერატორი გადაამოწმებს მონაცემებს

//TODO: ბეჭდავს პასუხს(Yes/No)

}

void Operator() {

while (customersLeft > 0) {

//TODO: ელოდება სანამ მომხმარებელი ჩადგება რიგში

//TODO: მომხმარებლის გამოძახება და ლოდინი სანამ მომხმარებელი შეავსებს ფორმას

int res = checkApplication();//აბრუნებს შედეგს - 1 თანხმობა, 0 უარყოფა

//TODO: მომხმარებლისთვის პასუხის დაბრუნება

}

//TODO: ბეჭდავს მის მიერ დამტკიცებული სესხების რაოდენობიის

}

int main() {

//მომხმარებლების ნაკადების გაშვება

for (int i=0; i<NUM\_CUSTOMERS; i++) {

pthread\_t \* t = malloc(sizeof(pthread\_t));

pthread\_create(t, NULL, Customer, NULL);

}

//ოპერატორების ნაკადების გაშვება

for (int i=0; i<NUM\_OPERATORS; i++) {

pthread\_t \* t = malloc(sizeof(pthread\_t));

pthread\_create(t, NULL, Operator, NULL);

}

pthread\_exit(0);

}