***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 აბსტრაქციები | | |
| ***ლექტორი*** | გ. ბოჭორიშვილი | | |
| ***კურსი*** | I | | |
| ***ჯგუფი*** |  | | |
| ***გამოცდის ფორმა*** | ღია წიგნი | | |
| ***გამოცდის***  ***ხანგრძლივობა*** | 4:00 საათი | | |
| ***მაქსიმალური ქულა*** | 240 | | |
| ***სტუდენტის სახელი და გვარი:*** | | | |

სახელი: ქულა:

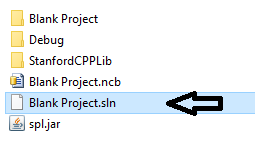
ფინალური გამოცდა (240 ქულა)

პროგრამირების აბსტრაქციები

13 ივლისი

**შენიშვნა: გამოცდიდან სტუდენტის მოხსნა ნებისმიერი მიზეზით ავტომატურად ნიშნავს, საგანში ფეილს.**

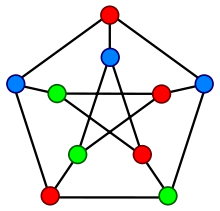
**შეასრულეთ შემდეგი ინსტრუქციები, წინააღმდეგ შემთხვევაში შესაძლოა თქვენი ნაშრომი არ შეფასდეს.**

1. **ჩამოტვირთეთ საგამოცდო საკითხები და პროექტის არქივი თქვენს დესკტოპზე, ამოაარქივეთ არქივი.**
2. **მაუსის Double Click ით გახსენით Deskto\midterm 2\project\Blank Project.sln, გაიხსნება Visual Studio.  
   **
3. **თუ Visual Studio ში არ გიჩანთ ფაილები, View მენიუში მონიშნეთ Solution Explorer პუნქტი**
4. **თითოეული ამოცანისთვის დაგხვდებათ შესაბამისი \*.cpp და \*.h ფაილები, სადაც უნდა დაწეროთ თქვენი ამოხსნა. main.cpp ფაილში წერია ამოცანების ტესტებზე შემოწმება. ფუნქციის პროტოტიპის შეცლა არ შეიძლება. შეგიძლიათ შემოიტანოთ დამხმარე ფუნქციები.**
5. **ამოხსნის დასრულების შემდეგ დაარქივეთ „Deskto\final\problems” ფოლდერი და არქივს დაარქვით სახელად თქვენი მეილის პრეფიქსი, მაგალითად gboch10.rar**
6. **ვებ ბრაუზერში გახსენით მისამართი** [http://192.168.210.5](http://192.168.210.5/) **და ატვირთეთ არქივი.**

**ამოცანა 1. გრაფის გაფერადება (70 ქულა)**

მოცემული გაქვთ არა მიმართული გრაფი და რიცხვი m(m<=6). დაადგინეთ შესაძლებელია თუ არა გრაფის წვეროების მაქსიმუმ m განსხვავებულ ფერში გაფერადება, ისე რომ არცერთი ორი ერთმანეთთან დაკავშირებული წვერო არ იყოს ერთ ფერში. გრაფის გაფერადება ნიშნავს, გრაფის თითოეული წვეროსთვის რაიმე ფერის მინიჭებას.

მაგალითად სურათზე მოცემული გრაფი შესაძლებელია გაფერადდეს 3 ფერის საშუალებით:



bool graphColoring(bool graph[V][V], int m);

დაწერეთ **graphColoring** ფუნქციის რელიზაცია. ფუნქიას გადაეცემა მატრუიცის სახით წარმოდგენენილი გრაფი და რიცხვი m. V არის გრაფში წვეროების რაოდენობა. მატრიცის [i][j] წევროს მნიშვნელობა არის true თუ i წვერო დაკავშირებულია j წვეროსთან პირდაპირი გზით, წინააღმდეგ შემთხვევაში უჯრა იქნება false.

ფუნქციამ უნდა დააბრუნოს true მხოლოდ იმ შემთხვევაში თუ გრაფის გაფერადება შესაძლებელია ზემოთ აღწერილი პირობებით. გრაფში წვეროების რაოდენობა არ აღემატება 15 ს, ტესტის დრო არ უნდა აღემატებოდეს 30 წამს.

**ამოცანა 2. ხე თუ გრაფი (70 ქულა)**

მოცემული გაქვთ არა მიმართული გრაფი დაწერეთ პროგრამა რომელიც დაადგენს გრაფი არის თუ არა ხე.

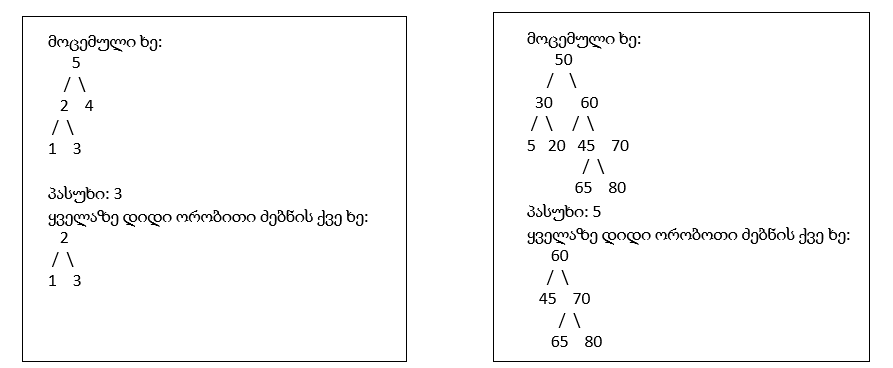
შეგახსენებთ გრაფი არის ხე თუ სრულდება ორი პირობა:

1. გრაფი არ შეიცავს ციკლს
2. გრაფი არის ბმული(ნებისმიერ ორი წვეეროსთცის არსებობს დამაკავშირებელი გზა)

დაწერეთ isTree ფუნქციის რეალიზაცია, რომელსაც გადაეცემა გრაფის სტრუქტურა და აბრუნებს true ს მხოლოდ იმ შემთხვევაში როცა გრაფი არის ხე. ფუნქცია უნდა მუშაობდეს O(n\*n) დროში, სადაც n არის წვეორების რაოდენობა.

**ამოცანა 3. უდიდესი ორობითი ქვე ხე (100 ქულა)**

თქვენი ამოცანაა დაწეროთ ფუნქცია რომელიც მოცემულ ორობით ხეში იპოვის უდიდეს ორობითი ძებნის ქვე ხეს. ორობითი ძებნის ხის ზომა განისაზღვრება მასში შემავალი ელემენტების რაოდენობით. ქვემოთ იხილეთ მაგალითები:



არ გაქვთ უფლება გამოიყენოთ მასივი და მონაცემთა სტრუქტურები(Vector, Stack, Queue …), გამოყენების შემთხვევაში ამოცანა განულდება.

დაწერეთ int largestBST(node \*root) ფუნქციის რეალიზაცია, ფუნქციას გადაეცემა ორობითი ხის root ელემენტის მისამართი და აბრუნებს მასში არსებული უდიდესი ორობითი ძებნის ხის ზომას.

ფუნქცია უნდა მუშაობდეს O(n) დროში, თუმცა თუ ასეთ ალგორითმს ვერ მოიფიქრებთ, დაწერეთ ნაკლებად ეფექტური ალგორითმით, ამით თქვენ გექნებათ ტესტების 40% ის აღების შანსი. გადმოცემულ ხეში არ იქნება ორი ერთანირი რიცხვი. ქვეხე არის საწყისი ხის რომელიმე ელემენტით და მის ქვემოთ არსებული ყველა ელემენტით(ფოთლების ჩათვლით) შედგენილი ხე.