***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 13:00 – 15:00

შეასრულეთ შემდეგი ინსტრუქციები, წინააღმდეგ შემთხვევაში შესაძლოა თქვენი ნაშრომი არ შეფასდეს.  
1. ჩამოტვირთეთ paradigms-exam2 ფოლდერი თქვენს დესკტოპზე. მასში უნდა იყოს 2 ფოლდერი assembly და disjoint.

2. ცვლილებები შეიტანეთ დავალების პირობით მითითებულ ფაილებში.

3. დააარქივეთ paradigms-exam2 ფოლდერი, არქივს სახელად დაარქვით გამოცდის ნომერი და თქვენი მეილის პრეფიქსი, მაგალითად 2gboch12.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. ასემბლერი (40 ქულა)**

დაწერეთ ასემბლის კოდი b მეთოდისთვის ცალკე გამოყავით თითოეული ხაზის შესაბამისი კოდი.

int printf(int x, ...);

**class** **bla**{

int b(int z, bla m){

int i = m.x[i];

**if** (i<x[i]){

printf(temp[1]->temp[0]->x[1], i, i);

}

**return** m.temp[1]->x[2];

}

bla \* temp[2];

int x[3];

};

ამოხსნა დაწერეთ assembly ფოლდერის assembly.cpp ფაილში.

**ამოცანა 2. განცალკევებული სიმრავლეები (80 ქულა)**

თქვენი ამოცანაა განცალკევებული სიმრავლეების სტრუქტურის რეალიზაცია. ორ სიმრავლეს ეწოდება განცალკევებული თუკი მათში არ არის არც ერთი ერთნაირი ელემენტი. მაგალითად სიმრაველეები (1,2,3), (10,12) და (4,6) განცალკევებული სიმრავლეებია. დაწერეთ სტრუქტურა disjointsets რომელიც შეინახავს განცალკევებულ სიმრავლეებს და ექნება შემდეგი ფუნქციები:

**typedef struct disjointsets{**

//თავად მოიფიქრეთ სტრუქტურის აღწერა

**};**

**void DisjointSetsNew(disjointsets \*ds);** - ინიციალიზაციას უკეთებს disjointset სტრუქტურას

**void DisjointSetsAdd(disjointsets \*ds, hashset \*s);** - ამ მეთოდის საშუალებით შეგვიძლია დავამატოთ ახალი სიმრავლე ჩვენს სიმრავლეებს. თუკი ახალი სიმრავლე ისეთია რომ არ აქვს თანაკვეთა არც ერთ უკვე არსებულ სიმრავლესთან მაშინ ის ჩვეულებრივად ემატება სიმრავლეებში მაგალითად:

[(1,2,3), (10,12), (4,6)] + (7,8) => [(1,2,3), (10,12), (4,6), (7,8)]

ხოლო თუკი დამატებულ სიმრავლეს აქვს თანაკვეთა არსებულ სიმრავლეებთან მაშინ ხდება ამ სიმრავლეების გაერთიანება, იმისათვის რომ საბოლოოდ ისევ განცალკევებული სიმრავლეები მივიღოთ. მაგალითად:

[(1,2,3), (10,12), (4,6)] + (1,6,9) => [ (1,2,3,6,9,4), (10,12)]

რადგან სიმრავლეს (1,6,9) თანაკვეთა ჰქონდა სიმრავლეებთან (1,2,3) და (4,6) ისინი ყველანი გაერთიანდნენ რათა ჩვენს სტრუქტურაში არსებული სიმრავლეები ისევ განცალკევებულები დარჩნენ.

**hashset \* DisjointSetsFind(disjointsets \*ds, void \* elemAddr);** - ამ ფუნქციის საშუალებით შეგვიძლია მოვძებნოთ ელემენტი ჩვენს სტრუქტურაში. ფუნქცია ეძებს ჩვენი განცალკევებული სიმრავლეებიდან თუ კონკრეტულად რომელშია გადმოცემული ელემენტი და აბრუნებს იმ სიმრავლეზე მიმთითებელს. თუკი ელემენტი არც ერთ სიმრავლეში არ მოიძებნა მაშინ ფუქნცია აბრუნებს NULL-ს.

**void DisjointSetsDispose(disjointsets \*ds);** - ასუფთავებს disjointset სტრუქტურის მიერ დაკავებულ მეხსიერებას.

ამოცანის გადასაჭრელად უნდა გამოიყენოთ hashset და vector სტრუქტურები. ასევე შეგიძლიათ ჩათვალოთ რომ ფუნქცია void SetUnion(hashset \*a, const hashset \*b) უკვე დაწერილია. SetUnion ფუნქციას გადაეცემა ორი სიმრავლე და ის ამ ორი სიმრავლეს აერთიანებს. ფუნქცია პირველ სიმრავლეში ჩაამატებს მეორე სიმრავლის ყველა ელემენტის.

შეზღუდვები:

თუ ჩავთვლით რომ HashSetEnter და HashSetLookup ფუნქციები O(1) დროში სრულდება, მაშინ

DisjointSetsNew უნდა სრულდებოდეს O(1) დროში

DisjointSetsAdd უნდა სრულდებოდეს O(n) დროში სადაც n სიმრავლეებში არსებული ელემენტების ჯამური რაოდენობაა

DisjointSetsFind უნდა სრულდებოდეს O(m) დროში სადაც m სტრუქტურაში სიმრავლეების რაოდენობაა

DisjointSetsDispose უნდა სრულდებოდეს O(n) დროში სადაც n სიმრავლეებში არსებული ელემენტების ჯამური რაოდენობაა

კომპილაციისთვის: gcc vector.c hashset.c hashsetunion.c disjointsets.c test.c

გაშვება: a.exe