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:

შუალედური გამოცდა
პარადიგმებში
2017, 24 ოქტომბერი 14:40 – 16:40

1 50 ქულა	2 70 ქულა	სულ

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

1. ჩამოტვირთეთ problems ფოლდერი თქვენს დესკტოპზე. მასში უნდა იყოს 2 ფოლდერი problem1 და problem2. თითოეულში კი შესაბამისი ფაილები.
2. ცვლილებები შეიტანეთ დავალების პირობით მითითებულ ფაილებში.
3. ის ფაილები, რომელშიც ცვლილებები შეიტანეთ დააარქივეთ, არქივს სახელად დაარქვით თქვენი მეილის პრეფიქსი, მაგალითად gboch12.rar. არქივში უნდა იყოს მხოლოდ 2 ფაილი
    1. find_score.c
    2. spell_correct.c
4. ვებ ბრაუზერში გახსენით მისამართი http://192.168.210.5 და ატვირთეთ არქივი.

command prompt-ის გამოსაყენებლად

1. დააჭირეთ windows ღილაკს ეკრანის მარცხენა ქვედა კუთხეში
2. ძებნის ფანჯარაში აკრიბეთ command prompt
3. დააკლიკეთ მაუსი command prompt-ის იკონს.
4. ფოლდერში ინფორმაციის ნახვისთვის გამოიყენეთ ბრძანება DIR(იგივე ls)
5. ფოლდერის შეცვლისთვის გამოიყენეთ cd

ამოცანა 2. ბოლოს (70 ქულა)

თუ შეგიმჩნევიათ უმრავლესობა ტექსტური რედაქტორებისა ინახავს ბოლოს დარედაქტირებული ფაილების სიას, რათა მარტივად შევძლოთ მათი ხელთავიდან გახსნა. ბოლოს დარედაქტირებული ფაილების სიაში ორჯერ არასდროს არ გვხვდება ერთი და იგივე ფაილი. სია ყოველთვის დალაგებულია და თავში არის ის ფაილი რომელიც ბოლოს დავარედაქტირეთ.
თქვენი ამოცანაა დაწეროთ ჯენერიქ სტრუქტურის რეალიზაცია რომელიც დაგვეხმარება მსგავსი ტიპის ინფომმაციის შენახვაში.

ქვემოთ იხილეთ RecentlyList სტრუქტურის ინტერფეისის აღწერა:

```
void RecentlyListNew(  RecentlyList * list,
                       size_t elemSize,
                       int (*cmp)(void *, void *),
                       void(*freeFn)(void *));
```

ფუნქციის საშუალებით ხდება სტრუქტურის ინიციალიზაცია, გადაეცემა სტრუქტურის მიმთითებელი, ელემენტის ზომა ბაიტებში, ორი ელემენტის შედარების ფუნქცია და ელემენტის გასუფთავების ფუნქცია. freeFn შესაძლოა NULL იყოს, ასეთ შემთხვევაში ელემენტი გასუფთავებას არ საჭიროებს. სტრუქტურას ინიციალიზაციის შემდეგ უნდა ჰქონდეს გამოყოფილი ზუსტად **INIT_ALLOC_LEN** ელემენტისთვის ადგილი.

```
void RecentlyListAdd(RecentlyList * list, void * e);
```

სიის თავში ამატებს ახალ ელემენტს, თუ **e** მნიშნელობის მქონე ელემენტი უკვე არსებობს სტრუქტურაში, მაშინ მან უბრალოდ უნდა გადმოინაცვლოს სიის თავში. შესაბამისად სიაში არასდროს არ უნდა იყოს ერთი და იგივე მნიშვნელობის მქონე ელემენტი.

```
void RecentlyListGet(RecentlyList * list, size_t indx, void * e);
```

e მისამართზე უნდა ჩაწეროს **indx** ნომრის მქონე ელემენტი, ჩათვალეთ რომ არასწორი ინდექსი არ გადმოეცემა ფუქნციას.

```
void RecentlyListRemove(RecentlyList * list, size_t indx);
```

ფუნქციამ უნდა წაშალოს სიიდან **indx** ნომრის მქონე ელემენტი.

```
size_t RecentlyListSize(RecentlyList * list);
```

ფუნქცია აბრუნებს სტრუქტურაში შენახული ელემენტების რაოდენობას

```
void RecentlyListDispose(RecentlyList * list);
```

ასუფთავებს სტრუქტურის და ელემენტების მიერ დაკავებულ ადგილს