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 1. მეფი მეხსიერებაში(50 ქულა)

პირველი კურსის სტუდენტების შუალედური გამოცდების შედეგები მოცემულია მეფის სახით. თქვენი ამოცანაა მოძებნოთ და დააბრუნოთ გადმოცემული სტუდენტის ქულა. მეფში გასაღებები სახელებია ხოლო მნიშვნელობები კი

ქულები. ეს მეფი პირდაპირ მეხსიერებაშია შენახული რათა ნაკლები ადგილი დაიკავოს. მეხსიერებაში ერთმანეთის მიყოლებით წერია სტუდენტის სახელი, მერე სტუდენტის ქულა, მერე სხვა სტუდენტის სახელი, სხვა სტუდენტის ქულა და ა.შ.
სტუდენტის სახელი შეიძლება იყოს ერთი ან მეტი ბაიტი სიგრძის. ჩათვალეთ, რომ სტუდენტების სახელები შედგება მხოლოდ დიდი ლათინური სიმბოლოებისგან ‘A’-‘Z’ და მათში არ გვხვდება არც ერთი სხვა სიმბოლო. მეხსიერების დაზოგვის მიზნით ეს სახელები(სტრინგები) სიმბოლო ‘\0’-ით არ ბოლოვდება. სახელი ბოლოვდება სტუდენტის ქულით.
როგორ გავარკვიოთ რამდენი სიმბოლოსგან შედგება სტუდენტის სახელი? როგორც მოგეხსენებათ დიდი სიმბოლოების ნომრები ASCII ცხრილში არ აღემატება 100-ს(‘A’ არის 65). შესაბამისად მათი ბიტური ჩანაწერი, რომ ვნახოთ მათი პირველი ბიტი ყოველთვის იქნება 0. ჩვენს შემთხვევაში სახელები(სტრინგები) ბოლოვდება ისეთი ბაიტით, რომელის პირველი ბიტიც 1-იანია. ანუ სტრინგის ბოლოს ნახვა თუ გინდათ უნდა ნახოთ პირველი შემხვედრი ისეთი ბაიტი, რომელის პირველი ბიტიც ერთია.
ამ “დაბოლოების ბაიტში” დანარჩენი 7 ბიტი აღნიშნავს სტუდენტის ქულას (ქულების მნიშვნელობებიც არ აღემატება 100-ს). ამ ქულის შემდეგ კი მოდის ისევ სხვა სტუდენტის სახელი-ქულა და ა.შ.

თქვენი ამოცანაა დაწეროთ ფუნქცია find_score რომელსაც გადაეცემა მეფის დასაწყისზე მიმთითებელი, სტუდენტების რაოდენობა მეფში, საძებნი სტუდენტის სახელი და რომელიც დააბრუნებს ამ სტუდენტის ქულას.
int find_score(void * memory, int n, char * name)

დავუშვათ მეხსიერება გამოიყურება შემდეგნაირად:
01000001.01000010.01000011.10000101.01000001.11100000.01000011.10100001
A B C 5 A 96 C 33

თუკი გადმოგვეცემა ეს მეხსიერება, 3 და სტრინგი “A” ჩვენმა ფუნქციამ უნდა დააბრუნოს 96.

მოცემული გაქვთ main.c ფაილი, სადაც ტესტები წერია, find_score.h ფაილი სადაც ჰედერია ფუნქციის და find_score.c ფაილი სადაც თქვენი კოდი უნდა დაწეროთ. გადმოაკოპირეთ ეს ფაილები თქვენს კომპიუტერში, მხოლოდ find_score.c ფაილში შეიტანეთ ცვლილებები. არ გამოიყენოთ გლობალური ცვლადები. პროგრამა უნდა დაიწეროს C-ში.

კომპილაციისთვის შეგიძლიათ გამოიყენოთ ბრძანება
> gcc main.c find_score.c
მიიღებთ a.exe ფაილს, რომელიც შეგიძლიათ გაუშვათ

ამოცანა 2. სპელკორექტორი(70 ქულა)

თქვენი მიზანია სპელკორექტორის დაწერა ამისათვის გაქვთ ლექსიკონი რომელშიც ყველა სიტყვა წერია. უნდა დაწეროთ ფუნქცია რომელსაც გადმოეცემა ლექსიკონი, ლექსიკონში არსებული სიტყვების რაოდენობა, რაღაც წინადადება, წინადადებაში არსებული სიტყვების რაოდენობა და რომელიც ამ წინადადებას გადააქცევს სწორი, გამართული სიტყვებისგან შედგენილ წინადადებად.
ჩათვალეთ რომ:
1. ლექსიკონი არის C-სტრიგნგების ზრდადობით დალაგებული მასივი.
2. წინადადებაში არსებული სიტყვები ერთმანეთისგან გამოყოფილია ერთი სფეისით.
3. შეცდომები რომელიც წინადადებაში შეიძლიება იყოს არის მხოლოდ და მხოლოდ ერთი სახის - რომელიმე გამართულ სიტყვას ბოლოში აქვს მიწერილი ერთი სიმბოლო(რის შედეგადაც ის სიტყვა წერია შეცდომით). სხვა ტიპის შეცდომები წინადადებაში არ გვხვდება.
4. თქვენს ფუნქციას გადაეცემა წინადადების მისამართი. წინადადება კი ჩვეულებრივი C-სტრინგია. ანუ თქვენს ფუნქციას გადაეცემა char**.
5. უნდა შეცვალოთ მხოლოდ ის სიტყვები, რომლებიც არასწორად წერია. სწორად დაწერილ სიტყვებს შეცვლა არ სჭირდება.
6. თუკი, რომელიმე სიტყვა ერთი სიმბოლოსგან შედგება, ეს სიტყვა სწორია.
აუცილებელია რომ:
1. თქვენი პროგრამა მუშაობდეს ეფექტურად - O(logN)*m დროში სადაც N ლექსიკონში არსებული სიტყვების რაოდენობაა ხოლო m კი წინადადებაში არსებული სიტყვების რაოდენობა.
2. გამოიყენოთ სტანდარტული ორობითი ძებნის ფუნქცია და არ დაწეროთ თქვენი.
void * **bsearch**(const void *key, const void *base, size_t nel, size_t width,int (*compar) (const void *, const void *));
3. მეთოდის დასრულების შემდეგ sent უნდა უთითებდეს მეხსიერებაზე სადაც გასწორებული წინადადება წერია.
4. არ დატოვოთ გაუთავისუფლებელი მეხსიერება

void spellCorrect(char*[] lex, int n, char sent){**

მოცემული გაქვთ spell_correct.h, spell_correct.c და main.c ფაილები. გადმოაკოპირეთ ისინი თქვენს კომპიუტერში. ცვლილებები შეგიძლიათ შეიტანოთ მხოლოდ spell_correct.c ფაილში.

კომპილაციისთვის გამოიყენეთ ბრძანება
> gcc main.c spell_corect.c
მიიღებთ a.exe ფაილს, რომელიც შეგიძლიათ გაუშვათ
> a.exe

ფუნქციების ჰედერები, რომლებიც შეიძლება დაგჭირდეთ ამოცანების გადაჭრისას:

```
void *memcpy(void *dest, const void *src, size_t n);
int memcmp(const void *s1, const void *s2, size_t n);
void *memmove(void *dest, const void *src, size_t n);
void *malloc(size_t size); void *realloc(void *ptr, size_t size);
void free(void *ptr);

size_t strlen(const char *s);
char *strcpy(char *dest, const char *src);
int strcmp(const char *s1, const char *s2);
char *strncpy(char *dest, const char *src, size_t n);
char *strdup(const char *s);
char *strcat(char *dest, const char *src);
```