***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:

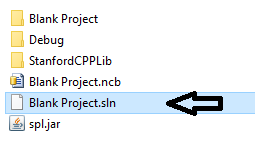
ფინალური გამოცდა (240 ქულა)

პროგრამირების აბსტრაქციები

6 სექტემბერი

**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 „Desktop\final fx\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. ბმულ სიად გადაქცევა (60 ქულა)**

განვიხილოთ შემდეგი სტრუქტურა:

struct node {

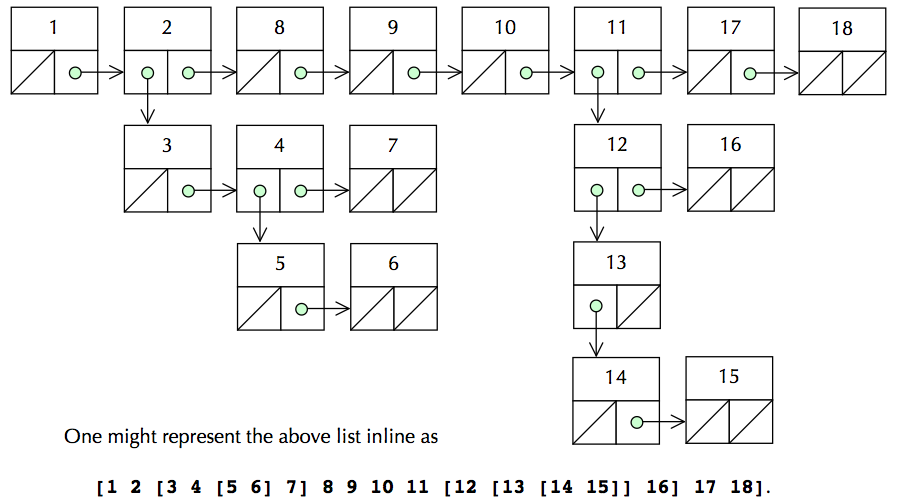
int value;

node\* down;

node\* next;

};

ეს სტრუქტურა შეიძლება შემდეგნაირად წარმოვადგინოთ

****

თქვენი ამოცანაა დაწეროთ **flattenList** მეთოდი რომელიც საშუალებას მოგვცემს ეს სტრუქტურა გადავაქციოთ ჩვეულებრივ ინტეჯერების ბმულ სიად, ის რომ ამ ელემენტის “ქვედა” სია(down) აღმოჩნდეს თვითონ ამ ელემენტსა და მის “შემდეგ”(next) ელემენტს შორის. მაგალითში მოყვანილი სტრუქტურა გარდაიქმნება შემდეგ სიად:

**[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18]**

მეთოდის იმპლემენტაციამ არ უნდა შეცვალოს შემომავალი სტრუქტურა და უნდა დააბრუნოს იგივე ტიპის სტუქტურა ოღონდ მისი ყველა down ელემენტი უნდა იყოს NULL.

**node\* flattenList(node\* list){**

არ გაქვთ Vector, Stack, Queue ან სხვა მზა მონაცემთა სტრუქტურის გამოყენების უფლება, ამოხსნა უნდა მუშაობდეს O(N\*N) დროში, სადაც N არის სიაში ელემენტების რაოდენობა. მოცემულ სტრუქურაში არ იქნება ციკლი და პირველი ემენეტიდან ნებისმიერ ელემენტამდე მხოლოდ ერთი გზით შეიძლება მისვლა.

**ამოცანა 2. კომბინაციები (90 ქულა)**

მოცემული გაქვთ დადებითი რიცხვი n. თქვენი ამოცანაა დათვალოთ 2\*n სიგრძის ყველა ისეთი კომბინაცია, რომლებშიც თითოეული რიცხვი 1 დან n მდე გვხვდება ზუსტად 2 ჯერ და მათ შორის დაშორება არის ამ რიცხვის მნიშვნელობის ტოლი ან ერთით ნაკლები.

მაგალითად n=2 ისთვის, ერთიანები შეიძლება იყვნენ განთავსებული ერთმანეთის გვერდიგვერდ ან ერთი უჯრით დაშორებით. ხოლო ორიანები უნდა იყვნენ ან ერთუ უჯრით დაშორებული ან ორით, შესაბამისად სულ სამი ვარიანტი გვაქვს:

1 2 1 2

2 1 2 1

2 1 1 2

n = 3 ისთვის პასუხი არის 6:

1 3 1 2 3 2

3 1 2 1 3 2

3 1 1 2 3 2

2 3 1 2 1 3

2 3 2 1 3 1

2 3 2 1 1 3

ასევე n = 4 ისთვის არის 32

დაწერეთ int allCombinations(int n) ფუნქციის რეალიზაცია, რომელიც აბრუნებს ზემოთ აღწერილი კომბინაციების რაოდენობას გადაცემული n ისთვის.

**ამოცანა 3. თამაში (90 ქულა)**

მოცემული გაქვთ გველების და კიბეების მართკუთხა ფორმის სათამაშო დაფა, უჯრები გადანომრილია 1 დან N(1 < N < 10000) მდე, სურათზე ნაჩვენებია მაგალითი.

[](http://www.geeksforgeeks.org/wp-content/uploads/snakesladders.jpg)

თავიდან მოთამაშე იმყოფება პირველ უჯრაზე. ყოველ სვლაზე აგორებს კამათელს და გადადის უჯრაზე ნომრით curr + x, სადაც curr არის უჯრის ნომერი სადაც იმყოფებოდა მოთამაშე, ხოლო x არის გაგორებული რიცხვი, 1 დან 6 მდე.

თუ გაგორების შემდეგ მოთამაშე მოხვდა უჯრაზე სადაც არის კიბის ქვედა ბოლო, ავტომატურად გადადის უჯრაზე სადაც არის ამავე კიბის ზედა ბოლო. თუ მოთამაშე მოხვდება უჯრაზე სადაც არის გველის თავი, მაშინ ავტომატურად გადადის უჯრაზე სადაც არის იგივე გველის კუდის ბოლო.

თქვენი ამოცანაა იპოვოთ მინიმუმ რამდენი გაგორებაა საჭირო იმისათვის რომ პირველი უჯრიდან მოთამაშე მოხვდეს ბოლო უჯრაზე.

მაგალითად ზემოთ მოცემული დაფისთვის პასუხი არის 3 გაგორება.

* პირველ გაგორებაზე უნდა გავაგოროთ 2, იმისათვის რომ მოვხვდეთ მესამე უჯრაზე, რაც ნიშნავს 22 უჯრაზე გადასვლას.
* შემდეგ უნდა გავაგოროთ 6 და გადავიდეთ 28 -ე უჯრაზე
* ბოლოს უნდა გავაგოროთ 2 და გადავალთ ბოლო უჯრაზე.

იგივე დაფაზე 3 გაგორებაში სხვა კომბინაციებითაც შეიძლება ბოლო უჯრამდე მისვლა, მაგალითად (2, 2, 6), (2, 4, 4), (2, 3, 5).

დაწერეთ **int getMinDiceThrows(int move[], int N)** ფუნქციის რეალიზაცია. ფუნქციას გადაეცემა N ელემენტიანი მასივის სახით წარმოდგენილი დაფა და რიცხვი N, დაფაზე უჯრების რაოდენობა.

* თუ i ნომრის მქონე უჯრაზე არც გველის თავია და არც კიბის ქვედა ბოლო move[i] არის -1-ის ტოლი
* თუ i ნომრის მქონე უჯრაზე არის გველის თავი/კიბის ქვედა ბოლო, მაშინ move[i] არის იმ უჯრის ნომერი სადაც მდებარეობს გველის კუდი/კიბის ზედა ბოლო

გაითვალისწინეთ დაფის მასივის სახით წარმოდგენაში უჯრების გადანომრვა იწყება 0 დან, შესაბამისად სურათზე ნაჩვენებ უჯრას ნომრით 1 შეესაბამება მასივის ნულოვანი ელემენტი. Main.cpp ფაილში არსებული ნაწილი ტესტებისთვის ქვემოთ მოცემულია დაფის სურათი, რაც დაგეხმარებათ პირობის გააზრებაში.

თქვენი ამოხსნა უნდა მუშაობდეს O(N) დროში, თუცა O(N\*N) დროში მომუშავე ალგორითმიც გაივლის ბევრ ტესტს.

ჩათვალეთ რომ

* პირველ უჯრაზე არასდროს არ იქნება კიბის ბოლო ან გველის თავი
* დაფა ისეთია რომ ყოველ სვლაზე მაქსიმუმ ერთიხელ შეიძლება კიბის/გველის გამო გადასვლა. ანუ არ იქნება შემთხვევა როცა უჯრაზე ერთდროულად გველის კუდი და გველის თავია ან კიბის ზედა ბოლო და ქვედა ბოლო.

პირველი ტესტის დაფა, პასუხი 3 (2, 6, 2)

[](http://www.geeksforgeeks.org/wp-content/uploads/snakesladders.jpg)

მეორე ტესტის დაფა, პასუხი 3 (2, 5, 1)



მესამე ტესტის დაფა, პასუხი 4 (3, 2, 6, 2)

