**ორი პრაქტიკული დავალება (თითო ფასდება 10 ქულით) ატვირთეთ ალგორითმის კოდი ან კოდის ლინკი (ონლაინ კომპილერის შემთხვევაში)**

1. მოცემულია A[10] მასივი. მასივი შეავსეთ რანდომად [-10;20] შუალედში. გამოითვალეთ მასივის კენტინდექსიანი მაქსიმალური წევრი და ლუწინდექსიანი მინიმალური წევრი. იპოვეთ ამ რიცხვების უსჯ და უსგ. კენტინდექსიანი მაქსიმალური წევრი ჩაანაცვლეთ უსჯ-თი. დაბეჭდეთ მიღებული მასივი

<https://onlinegdb.com/E6jTTSzaG>

1. იპოვეთ ყველა ექვსნიშნა რიცხვი (ასეთი რიცხვები ჩაწერეთ რიცხვით ვექტორში), რომელთა პირველი სამი და ბოლო სამი ციფრი ტოლია. დაბეჭდეთ რიცხვები და მათი რაოდენობა

<https://onlinegdb.com/dSuHTFvR8>

**ორი ღია კითხვა. გააანალიზეთ კითხვის შინაარსი და უპასუხეთ ამომწურავად. (თითო ფასდება 2 ქულით)**

1. როგორ შეგიძლიათ ახსნათ ტერმინი ალგორითმების „ოპერაციული ღირებულება“

**პასუხი**

ალგორითმების განხილვისას ტერმინი „ოპერაციული ღირებულება“ ჩვეულებრივ ეხება

გამოთვლით რესურსებს, რომლებიც საჭიროა ალგორითმის ფარგლებში კონკრეტული

ოპერაციის შესასრულებლად. გამოთვლითი რესურსები შეიძლება იყოს დრო, სივრცე ან

სხვა რესურსები კონტექსტიდან გამომდინარე.

დროის სირთულე: საოპერაციო ღირებულება ხშირად ეხება ოპერაციის დროს, რომელიც

ზომავს ალგორითმის დასრულებისათვის საჭირო დროს მისი შეყვანის ზომის მიხედვით.

სივრცის სირთულე: ოპერაციული ღირებულება ასევე შეიძლება ეხებოდეს

ოპერაციისთვის საჭირო სივრცეს, იზომება მეხსიერების რაოდენობას, რომელიც

საჭიროა ალგორითმის მიერ პრობლემის გადასაჭრელად, მისი შეყვანის ზომის

მიხედვით. ის რაოდენობრივად განსაზღვრავს მეხსიერების რაოდენობას, რომელიც

საჭიროა ცვლადების, მონაცემთა სტრუქტურების და სხვა რესურსების შესანახად,

რომლებიც გამოიყენება ალგორითმის შესრულების დროს.

სხვა რესურსები: კონკრეტული კონტექსტიდან გამომდინარე, ოპერაციის ღირებულება

შეიძლება ასევე ეხებოდეს სხვა რესურსებს, როგორიცაა ქსელის გამტარუნარიანობა,

ენერგიის მოხმარება ან ალგორითმის მიერ მოთხოვნილი სხვა შესაბამისი რესურსები.

ოპერაციათა ღირებულება დამოკიდებულია იმ სიმრავლეთა

ზომებზე, რომელთა მიმართაც ისინი გამოიყენებიან.

1. რიგზე მუშაობს ფუნქციები front() და back() ახსენით ამ ფუნქციების მუშაობა

front()-მიმართვა სათავეში მყოფ ელემენტზე

back()-მიმართვა ბოლოში მყოფ ელემენტზე

1. ახსენით push, top, pop და რიგზე მუშაობის ფუნქციები რას აკეთებს. ახსენით განსხვავება

push()-ელემენტის ჩამატება რიგის ბოლოში

pop ()-ელემენტის წაშლა რიგის სათავეში, ყოველთვის იშლება სათავეში მყოფი ელემენტი, ხოლო მისი მომდევნო ელემენტი ხდება ახალი სათავე

size()-აბრუნებს რიგში არსებულ ელემენტთა რაოდენობას

empty()-ბულის ტიპის ფუნქცია, რომელიც გვატყობინებს არის თუ არა რიგი ცარიელი. მისი მნიშვნელობა შეიძლება იყოს მხოლოდ TRUE ან FALSE.

1. რა განსხვავებაა მეხსიერების გადანაწილების სისტემებში მასივსა და ვექტორისთვის

მეხსიერების განაწილება: მასივისთვის გამოყოფილია სტეკზე ან გროვაზე,

ხოლო ვექტორისთვის ყოველთვის გამოყოფილია გროვაზე. ეს ნიშნავს, რომ

მასივებს აქვთ მეხსიერების ფიქსირებული მოცულობა, რომელიც გამოყოფილია

კომპილაციის ან გაშვების დროს, ხოლო ვექტორის მოცულობა ცვალებადია.

1. რა განსხვავებაა მასივის და ვექტორის ელემენტების წვდომის სისტემებში

წვდომა: მასივის ელემენტებზე წვდომა შესაძლებელია [] ოპერატორის

გამოყენებით, ხოლო ვექტორის ელემენტებზე წვდომა შესაძლებელია

როგორც [] ოპერატორის, ისევე .at() ფუნქციის გამოყენებით. .at()

ფუნქცია უზრუნველყოფს საზღვრების შემოწმებას, ხოლო [] ოპერატორი არა.

1. როგორ არის შესაძლებელი იტერაციის პროცესების რეალიზება(ვექტორებზე, მასივებზე)

იტერაცია: მასივების და ვექტორების იტერაცია შესაძლებელია დიაპაზონზე

დაფუძნებული for loop-ის გამოყენებით, ვექტორებზე იტერაცია კი ასევე

შეიძლება იტერატორების გამოყენებით, რაც უზრუნველყოფს მეტ მოქნილობას

და შეიძლება გამოყენებულ იქნას ელემენტების ადვილად ჩასმისა ან

წაშლისათვის (იტერატორები, როგორც წესი, იქმნება კონტეინერის begin() და

end() წევრის ფუნქციების გამოყენებით, რომლებიც იტერატორს აბრუნებენ

შესაბამისად კონტეინერის/ვექტორის პირველ და ბოლო ელემენტებზე).

1. overflow ახსენით რა მოვლენაა და რა სიტუაციებში ხდება ასევე underflow

თუ top[S]=0, მაშინ სტეკი ცარიელია (is empty).

თუ top[S]=n, მაშინ სტეკში ელემენტის დამატების მცდელობისას

ხდება გადავსება (overflow), რადგან სტეკის ზომა ჩვენს მაგალითში

განსაზღვრულია n-ით.

სიმეტრიული სიტუაცია იქმნება ცარიელი სტეკიდან ელემენტის

ამოღების მცდელობისას (underflow)-გადაცარიელება”.

სტეკის გადავსება(overflow) - ამ მოვლენას ადგილი აქვს, თუ არის მცდელობა ელემენტის ჩამატებისა, როდესაც სტეკი სავსეა

სტეკის underflow - ამას მოვლენას ადგილი აქვს, როდესაც არის მცდელობა წაიშალოს ელემენტი

ცარიელი სტეკიდან

მასივი და ვექტორები

ზომა: მასივს აქვს ფიქსირებული ზომა, რომელიც მიეთითება გამოცხადებისას, ხოლო ვექტორის ზომა შეიძლება დინამიკაში შეიცვალოს ელემენტების დამატების ან წაშლისას.

ჩასმა და წაშლა: მასივში ელემენტების ჩამატება ან წაშლა საკმაოდ

რთული და შრომატევადია, რადგან ის მოითხოვს ყველა ელემენტის

გადატანას ჩასმის ან წაშლის წერტილიდან დაწყებული. ვექტორში

ელემენტების ჩასმა ან წაშლა ადვილად შესაძლებელია ნებისმიერ

პოზიციაზე, რადგან ვექტორში მეხსიერება უბრალოდ გადანაწილდება

საჭიროებისამებრ.

დასკვნისთვის: მასივები საუკეთესოა სიტუაციებში, სადაც ელემენტების

ნაკრების ზომა წინასწარ არის დაფიქსირებული და ცნობილი, ხოლო ვექტორები,

როგორც უფრო მოქნილი, შესაფერისია იმ სიტუაციებისთვის, სადაც

ელემენტების ნაკრების ზომა შეიძლება დინამიკაში შეიცვალოს.

ალგორითმის თვისებებია:

– დ ი ს კ რ ე ტ უ ლ ო ბ ა. ალგორითმი შედგება ცალკეული

ბრძანებებისაგან. სხვადასხვა ბრძანების შესრულება ერთი ბიჯის

ფარგლებში დაუშვებელია.

– გ ა ს ა გ ე ბ ო ბ ა. ალგორითმი გასაგებია შემსრულებლისათვის, თუ

ალგორითმის ყოველი ბრძანება შედის შემსრულებლის ბრძანებათა

სისტემაში.

– გ ა ნ ს ა ზ ღ ვ რ უ ლ ო ბ ა. ალგორითმში მკაცრადაა განსაზღვრული

ბრძანებების შესრულების რიგი; ყოველი ბრძანება ცალსახად აღწერს

შესასრულებელ მოქმედებას. მოქმედების დაწყებამდე მასში ყველა

ცვლად სიდიდეს აქვს მნიშვნელობა.

– შ ე დ ე გ ი ა ნ ო ბ ა. ალგორითმის შესრულებისას სასრული

რაოდენობის ბიჯების შემდეგ ამოცანის ამოხსნის შედეგი აუცილებლად

მიიღება.

არსებობს ალგორითმის გამოსახვის სამი

ძირითადი ფორმა:

სიტყვიერი;

ფსევდოკოდი, რომელსაც ესაჭიროება სპეციალური კოდის

გამოყენება;

ბლოკ-სქემური, რომელიც ძალზე მოხერხებულია თავისი

თვალსაჩინოების გამო.

წრფივ სიებზე შეიძლება შესრულდეს შემდეგი ოპერაციები:

1. მოვახდინოთ მიმართვა სიის k-ურ ელემენტზე მისი მინდვრების მნიშვნელობის შემოწმების და(ან) შეცვლის მიზნით;
2. ახალი ელემენტის ჩამატება უშუალოდ k-ური ელემენტის წინ ან შემდეგ;
3. k-ური ელემენტის წაშლა;
4. ერთ წრფივ სიაში ორი (ან მეტი) წრფივი სიის გაერთიანება;
5. წრფივი სიის გაყოფა ორ (ან მეტ) წრფივ სიად;
6. წრფივი სიის ასლის შექმნა;
7. წრფივ სიაში ელემენტების რაოდენობის განსაზღვრა;
8. ელემენტების დალაგება ამ ელემენტებით განსაზღვრული მინდვრების მნიშვნელობების ზრდადობით (ან კლებადობით);
9. ელემენტის მოძებნა რომელიმე მინდვრის მოცემული მნიშვნელობით.

ხშირად იყენებენ გამოთქმას, რომ ელემენტი ჩაიტვირთა (push) სტეკში ან

ამოიტვირთა სტეკიდან (pop).

რიგთან მიმართებაში გამოიყენება ტერმინი “დასაწყისი” (front) და “ბოლო”

(rear).

დროებითი ცვლადები მეხსიერების იმ ნაწილში თავსდება, რომელსაც სტეკი (stack)

ეწოდება.

მრავალი დროებითი ცვლადის გამოყენებამ შეიძლება “სტეკის გადავსების” შეცდომა

(stack overflow) გამოიწვიოს.

სტეკში დროებითი ცვლადის მიერ დაკავებული ადგილი თავისუფლდება შესაბამისი

ბლოკის დასრულების შემდეგ. ბლოკში ყოველი შესვლისას დროებით ცვლადს სტეკში

ხელახლა გამოეყოფა ადგილი.

არითმეტიკულ ოპერაციებს მივანიჭოთ პრიორიტეტები:

^ − მაღალი, / და \* − საშუალო, - და + − დაბალი.

გამოსახულების ინფიქსური ჩანაწერის გარდაქმნა პოსტფიქსურად:

იდენტიფიკატორი (ან რიცხვი) პირდაპირ გადაიწერება გამომავალ სტრიქონში ;

თუ სტეკი ცარიელია, ან სტეკის ზედაპირზე მოთავსებულ ოპერაციას ნაკლები პრიორიტეტიაქვს, ვიდრე განსახილველს, ოპერაცია თავსდება სტეკში;

ოპერაცია სტეკიდან გამომავალ სტრიქონში “გამოაძევებს” მისი ტოლი ან მეტი პრიორიტეტის მქონე ყველა ოპერაციას;

მარცხენა ფრჩხილი ყოველთვის თავსდება სტეკში;

მარჯვენა ფრჩხილი სტეკიდან გამომავალ სტრიქონში “გამოაძევებს” ყველა ოპერაციას

უახლოეს მარცხენა ფრჩხილამდე. თავად ფრჩხილები გამომავალ სტრიქონში არ გადაიწერება -შესაბამისი წყვილი უბრალოდ წაიშლება.

ყველა ოპერაციის ჩატარების შემდეგ სტეკში დარჩენილი რიცხვი წარმოადგენს

გამოსახულების მნიშვნელობას.

სტეკის ფუნქციები:

push() -ელემენტის ჩამატება სტეკში. ახლად დამატებული ელემენტი

ყოველთვის ხდება სტეკის სათავე, ხოლო ის ელემენტი კი,

რომელიც აქამდე იყო სტეკის სათავე, ხდება მისი მომდევნო.

pop()-ელემენტის წაშლა სტეკიდან. ყოველთვის იშლება სათავეში მყოფი

ელემენტი, ხოლო მისი მომდევნო ელემენტი ხდება ახალი სათავე.

top()-მიმართვა სათავეში მყოფ ელემენტზე.

size()-ეს ფუნქცია აბრუნებს სტეკში არსებულ ელემენტთა რაოდენობას.

empty()-ბულის ტიპის ფუნქცია, რომელიც გვატყობინებს არის თუ არა

სტეკი ცარიელი. მისი მნიშვნელობა შეიძლება იყოს მხოლოდ

TRUE ან FALSE.

რიგი

რიგი (queue) მონაცემთა აბსტრაქტული ტიპია,

რომელიც წარმოადგენს ელემენტთა სიას და ამ

ელემენტებისაგან წვდომა ხორციელდება მხოლოდ ორ

მათგანზე - სათავეში მოთავსებულ ელემენტზე და ბოლოში

მოთავსებულ ელემენტზე.

სხვაგვარად რომ ვთქვათ, რიგის მოქმედების პრინციპია

„პირველი მოვიდა, პირველი წავიდა“ (First In, First Out –

FIFO). რიგში მოთავსებული ელემენტების ინდექსაცია არ

ხდება.

რიგისათვის ელემენტის დამატების ოპერაციას დავარქვათ ENQUEUE,

ხოლო რიგიდან ელემენტის წაშლის ოპერაციას – DEQUEUE.

სტეკის მსგავსად წასაშლელი ელემენტი რიგშიც ცალსახად

განსაზღვრულია და ამიტომ ის კი არ გადაეცემა DEQUEUE პროცედურას,

არამედ ეს პროცედურა აბრუნებს მის მნიშვნელობას.

რიგს გააჩნია დასაწყისი ანუ თავი (head) და ბოლო (tail).

რიგისათვის დამატებული ელემენტი აღმოჩნდება მის ბოლოში (როგორც

უკანასკნელად მოსული მყიდველი ნამდვილ რიგში), ხოლო წასაშლელი

ელემენტი იმყოფება რიგის თავში (როგორც ყველაზე ადრე მოსული

მყიდველი).

თუ head[Q]=tail[Q], მაშინ რიგი ცარიელია

თავდაპირველად ყოველთვის head[Q]=tail[Q]=1.

ცარიელი რიგიდან ელემენტის წაშლის მცდელობა იწვევს underflow

(გადაცარიელება) შეცდომას,

თუ რიგი მთლიანადაა შევსებული, მისთვის ელემენტის დამატების

მცდელობა იწვევს overflow (გადავსება) შეცდომას.

(ქვემოთ მოყვანილ პროცედურებში ეს შეცდომები იგნორირებულია.

დეკი(deque) “double-ended-queue” ფრაზის შემოკლებული აღნიშნვაა, რომლის

ინგლისური თარგმანი ნიშნავს ორმხრივ რიგს.

დეკი კონტეინერი ძალიან ჰგავს კონტეინერ ვექტორს - ისევე როგორც ვექტორი, დეკიც

წარმოადგენს დინამიურ მასივს. განსხვავება ვექტორს და დეკს შორის ისაა, რომ

დეკში დინამიური მასივი ღიაა ორივე მხარეს. ეს საშუალებას იძლევა დეკს ძალიან

სწრაფად დაემატოს ახალი ელემენტი, როგორც დასაწყისში, ისე ბოლოში. ვექტორში კი

ელემენტის დამატება შესაძლებელია მხოლოდ მასივის ბოლოში.

დეკის ელემენტებზე წვდომა შეიძლება განხორციელდეს **იტერატორების** საშუალებით.

დეკის ფუნქციები

front()-მიმართვა დეკის სათავეზე

back()-მიმართვა დეკის ბოლოზე

push\_front()-ელემენტის ჩამატება დეკის სათავეში

push\_back()-ელემენტის ჩამატება დეკის ბოლოში

pop\_front()-ელემენტის წაშლა დეკის სათავეში

pop\_back()-ელემენტის წაშლა დეკის ბოლოში

size()-აბრუნებს დეკში არსებულ ელემენტთა რაოდენობას

Resize()-ფუნქციით შესაძლებელია დეკის ზომის გადიდება, მაგრამ არ არის შესაძლბელი დეკის ზომის შემცირება.

empty()-ბულის ტიპის ფუნქცია, რომელიც გვატყობინებს არის თუ არა დეკი ცარიელი. მისი

მნიშვნელობა შეიძლება იყოს მხოლოდ TRUE ან FALSE.

**ტესტური კითხვები (თითო ფასდება 0,5 ქულით)**

1. **“დასაწყისი” (front) და “ბოლო” (rear) აქვს წრფივი სიის ერთ ერთ სახეს:** რიგს; \*
2. **s.erase(s.begin(),s.begin()+4) ფუნქცია** \*s სტრიქონიდან წაშლის პირველიდან მეხუთემდე სიმბოლოებს; \*
3. **არგუმენტის გადაცემის რომელ ხერხს ვერ გამოვიყენებთ ფუნქციაში სტეკის (რიგის, დეკის) გადაცემისას:** pass-by-value; \*
4. **ელემენტებზე წვდომა შეიძლება განხორციელდეს იტერატორების საშუალებით:** დეკში; \*
5. **რიგი გამოიყენება:** **\***მონაცემთა ნაკადების  ბუფერიზაციის პროცესის უზრუნველსაყოფად, ინფორმაციული პაკეტების ნაკადის მოსაწესრიგებლად კომპიუტერულ ქსელებში; \*
6. **მოქმედებებს სიმრავლეებზე, რომლებიც ცვლიან სიმრავლეებს ეწოდებათ: \*** (modifying operations) ოპერაციები; \*
7. **Vector კლასზე განსაზღვრული max\_element ფუნქციის პროტოტიპია   
   Forward It   max\_element(ForwardIt first, ForwardIt last); რომელი მსჯელობაა სწორი \***ფუნქციის არგუმენტებია იტერატორები, დასაბრუნებელი ტიპიც იტერატორი;\*
8. **Vector კლასზე განსაზღვრული min\_element ფუნქციის პროტოტიპია   
   Forward It   min\_element(ForwardIt first, ForwardIt last); რომელი მსჯელობაა სწორი \***ფუნქციის არგუმენტებია იტერატორები, დასაბრუნებელი ტიპიც იტერატორი;\*
9. **str1.compare(str2); ფუნქცია მთელ რიცხვს. რომელი მსჯელობაა სწორი: \***თუ რიცხვი დადებითია str1 მეტია str2-ზე და რიცხვის მნიშვნელობა გვიჩვენებს პირველი განსხვავებული სიმბოლოს კოდებს შორის სხვაობას;\*
10. **თუ სტრიქონები წარმოდგენილია სიმბოლოების მასივების სახით.  მეხსიერების გამოყოფა შესაძლებელია რამდენიმე გზით. რომელი არ არის სწორი გზა? \***იტერაციული განაწილება; \*
11. **null ტერმინატორი მიუთითებს** **\***სტრიქონის დასასრულს
12. **რომელი მსჯელობაა სწორი სტრიქონების შეასახებ**

**\***კლასის ობიექტების გამოცხადებისას მეხსიერება გამოიყოფა გამოცხადების გაკეთებბის მომენტში და ობიექტი იგივდება იმ სახელთან, რომელიც განაცხადშია მითითებული

1. **მითითებული ობიექტის გამოყენება არ შეგვიძლია სტრიქონის კლასის ობიექტზე** d)^=\*

**\***ეს ხუთი შეგვიძლია = ; == ; > ; + ; += და პასუხი კარგად არ ჩანდა ფოტოზე

1. **სტრიქონის შექმნის მომენტში ცარიელ სტრიქონს თავიდანვე გამოეყოფა** **\***15 ბაიტი
2. **S.append(str,0,6) ფუნქცია \***აბრუნებს სახეშცვლილი სტრიქონს s-ს, რომლესაც ბოლოში დაემატება str-ის პირველი 6 სიმბოლო
3. **ფუნქცია str.empty()** **\***ამოწმებს ცარიელია თუ არა სტრიქონი
4. **Str.find(“s”,pos) ფუნქცია** **\***str სტრიქონში იპოვის a სიმბოლოს პირველ გამოჩენას pos პოზიციის შემდეგ
5. **Reverse(str.begin()+2, str.end()-4) ფუნქცია** **\***შეაბრუნებს str სტრიქონს დაწყებული მესამე პოზიციიდან ბოლოდან მე-4 პოზიაციამდე
6. **Str1.insert(8, str2) ფუნქცია** **\***Str2 სტრიქონს ჩაამატებს Str1 სტრიქონში მერვე პოზიციიდან
7. **Str1.replace(0, 4, str2) ფუნქცია** **\***Str1 სტრიქონის პირველ 4 პოზიციას ჩაანცვლებს Str2 სტრიქონი
8. **ორი სტრიქონის შედარების შედეგია** **\***მთელი ტიპის სიდიდე
9. **Sort(v.rbegin(), v.rend()) ფუნქცია ვექტორს დაალაგებს** **\***კლებადობის მიხედვით
10. **Count\_if(vec.begin(), vec.end()) მიმართვა ფუქნცია პრედიკატზე** **\***დაითვლის იმ ელემენტების რაოდენობას რომელზე პრედიკატი დააბრუნებს ჭეშმარიტ/მცდარ მნიშვნელობას. აქ ორივე ჭეშმარიტიც და მცდარიც ფიფქით იყო მონიშნული
11. **Count\_if(ForwardIt first, ForwardIt last, UnaryPredicate pred); \***ფუნქციის არგუმენტებია იტერატორები და ფუნქცია, დასაბრუნებელი - მთელი ტიპი
12. **Begin() და end() იტერატორები შესაბამისად მიმართავენ კონტეინერის** **\***პირველ და ბოლოს მომდევნო ელემენტებს.
13. **rend() და rbegin() იტერატორები შესაბამისად მიმართავენ კონტეინერის** **\***პირველის წინა და ბოლო ელემენტებს
14. **ალგორითმის ერთი ბიჯის ფარგლებში სხვადასხვა ბრძანების შესრულება დაუშვებელია - ალგორითმის ეს თვისებაა \***დისკრეტულობა
15. **ევკლიდეს ალგორითმი შეეხება \***უდიდესი საერთო გამყოფის პოვნას
16. **სიმრავლეები, რომლებიც იცვლებიან ალგორითმის მუშაობის პროცესში არის \***დინამიური სიმრავლეები
17. **დინამიური სიმრავლის ელემენტს წარმოადგენს ჩანაწერი, რომელიც შეიცავს სხვადასხვა მინდვრებს, ველს, რომელიც განკუთვნილია ელემენტის ცალსახად განსაზღვრისთვის ეწოდება \***გასაღები(key)
18. **მოქმედებებს სიმრავლეებზე, რომლებიც არ ცვლიან ამ სიმრავლეებს ეწოდებათ** **\*** (queris)მიმართვები
19. **მოქმედებებს სიმრავლეებზე, რომლებიც ცვლიან ამ სიმრავლეებს ეწოდებათ \*** (modifying operations)ოპერაციები
20. **სიმრავლეზე ჩატარებული ოპერაციების ღირებულება დამოკიდებულია \***იმ სიმრავლეების ზომებზე, რომელთა მიმართაც ისინი გამოიყენებიან
21. **წრფივი სიები მიეკუთვნება \***დინამიურ სიმრავლეებს
22. **წრფივ სიებზე არ არის შესაძლებებლი შემდეგი მოქმედებების შესრულება \***სიის შუაში მდგომი ელემენტის ბეჭდვა
23. **წრფივ სიებზე არ არის შესაძლებებლი შემდეგი მოქმედებების შესრულება \***ერთი ელემენტის რომელიმე მინდვრისთვის გასაღების მნიშვნელობის მინიჭება
24. **სტეკი ესაა წრფივი სია, რომელშიც \***რომელშიც ჩასმის, წაშლის და მნიშვნელობებზე მიმართვის ოპერაციები ხდება მხოლოდ სიის ერთ ბოლოზე
25. **რიგი ანუ ცალმხრივი რიგი - ესაა წრფივი სია \***რომელშიც ჩასმის ყველა ოპერაცია ხორციელდება ერთ ბოლოზე, ხოლო წაშლის (და როგორც წესი მნიშვნელობებზე მიმართვის) ოპერაცია - მეორე ბოლოზე
26. **დეკი ანუ ორმხრივი რიგი ესაა წრფივი სია \***რომელშიც ჩასმის, წაშლის და მნიშვნელობებზე მიმართვის ოპერაციები ხდება სიის ორივე ბოლოზე
27. **სტეკის მუშაობის პრინციპია \***„ბოლო მოვიდა - პირველი წავიდა“(LIFO)
28. **რიგის მუშაობის პრინციპია \***„პირველი მოვიდა-პირველი წავიდა“(FIFO)
29. **სტეკთან დაკავშირებულია სამი ოპერაცია -ელემენტის ჩატვირთვა, წაშლა და ელემენტზე მიმართვა, \***სამივე ოპერაცია სრულდება 0(1) დროში
30. **სტეკი გამოიყენება \***ფუნქციის გამოძახების, ლოკალური ცვლადების მართვისთვის და ოპერაციების რიგის შესანარჩუნებლად
31. **რიგი გამოიყენება \***მონაცემთა ნაკადის ბუფერიზაციის პროცესის უზრუნველსაყოფად, ინფორმაციული პაკეტების ნაკადის მოსაწესრიგებლად კომპიუტერულ ქსელებში
32. **დეკი გამოიყენება \***დიაგრამის გადაკვეთის ალგორითმებში, ამოცანების სიის სამართავად, მეხსიერების მართვის, შემავალი და გამომავალი ბუფერების და სიმულაციურ სისტემებში
33. **ელემენტებზე წვდომა შეიძლება განხორციელდეს იტერატორების საშუალებით \***დეკში
34. **წრფივი სიის ერთერთ სახეში წაშლის ოპერაციის შესრულებისას პირველ ყოვლისა წაიშლება სიის ყველაზე „უმცროსი“ ობიექტი, ანუ ობოექტი, რომელიც ყველაზე ბოლო იქნა ჩასმული \***სტეკში
35. **წრფივი სიის ერთერთ სახეში ყველაზე ადრე წაიშლება „უფროსი“ ელემენტი, ანუ ის, რომელიც ყველაზე ადრე ჩაისვა \***რიგში
36. **სტეკთან დაკავშირებული ოპერაციები სრულდება \***სამივე 0(1) დროში
37. **ფუნქციის გამოძახების და ლოკალური ცვლადების მართვისთვის, ოპერაციების რიგის შესანარჩუნებლად გამოიყენება \***სტეკი
38. **არგუმენტების გადაცემის რომელ ხერხს ვერ გამოვიყენებთ ფუნქციაში სტეკის(რიგის, დეკის) გადაცემისას: \***pass-by-value
39. **არითმეტიკული გამოსახულების ინფიქსური ფორმიდან პოსტფიქსურ ფორმაზე გარდაქმნის დროს გამოსახულებაში აღარ ჩაიწერება \***გამხსნელი და დამხურავი ფჩხილები
40. **რომელი მსჯელობაა მართებული არითმეტიკული გამოსახულების ინფიქსური ფორმიდან პოსტფიქსურ ფორმაზე გარდაქმნის პროცესზე მსჯელობისას: \***მარჯვენა ფრჩხილი სტეკიდან გამომავალ სტრიქონში გამოაძევებს ყველა ოპერაციას უახლოეს მარცხენა ფრჩხილამდე
41. **პოსტფიქსური გამოსახულების შესრულების დროს თუკი გამოსახულებაში გვხვდება ოპერაციის ნიშანი (ვთქვათ, \* ) და სტეკის ზედაპირზეა x, ხოლო მის ქვემოთ y: \***სრულდება y \* x ოპერაცია. სტეკიდან გამოძევდება x-იც და y-იც და მათ ნაცვლად თავსდება შესრულებული ოპერაციის შედეგი; \*
42. **რომელი მსჯელობაა მართებული არითმეტიკული გამოსახულების ინფიქსური ფორმიდან პოსტფიქსურ ფორმაზე გარდაქმნის პროცესზე მსჯელობისას : \***თუ სტეკი ცარიელია, ან სტეკის ზედაპირზე მოთავსებულ ოპერაციას ნაკლები პრიორიტეტი აქვს, ვიდრე განსახილველს, ოპერაცია თავსდება სტეკში; \*
43. **რომელი მსჯელობაა მართებული არითმეტიკული გამოსახულების ინფიქსური ფორმიდან პოსტფიქსურ ფორმაზე გარდაქმნის პროცესზე მსჯელობისას : \***მარცხენა ფრჩხილი ყოველთვის თავსდება სტეკში
44. **რომელი გამოსახულებაა სწორი პოსტფიქსური გამოსახულების შესრულების შესახებ: \***იდენტიფიკატორის რიცხვითი მნიშვნელობა(ან რიცხვი) პირდაპირ გადაიწერება სტეკში
45. **წრფივი სიის ერთერთი სახე გამოიყენება დასაბეჭდად გამზადებული მონაცემთა ნაკადის ბუფერიზაციის პროცესის უზრუნველსაყოფად. ეს არის: \***რიგი
46. **წრფივი სიის ერთერთი სახე გამოიყენება კომპიუტერულ ქსელებში ინფორმაციული პაკეტების გადაცემის პროცესის მოსაწესრიგებლად. ეს არის: \***რიგი
47. **წრფივ სიაში ელემენტებზე წვდომა ხორციელდება მხოლოდ ორ ელემენტზე. ჩატვირთვა ხდება ბოლოდან ხოლო წაშლა სათავიდან. ეს სიაა: \*** რიგი
48. **სია რომელიც საშუალებას გვაძლევს ახალი ელემენტი დაემატოს როგორც დასაწყისში ასევე ბოლოში არის:** **\***დეკი
49. **მართებულია თუ არა ის აზრი რომ დეკის ელემენტებზე წვდომა შეიძლება განხორციელდეს იტერატორების საშუალებით:** **\***კი
50. **მართებულია თუ არა ის აზრი რომ სტეკის ელემენტებზე წვდომა შეიძლება განხორციელდეს იტერატორების საშუალებით:** **\***არა