1) STL ბიბლიოთეკა შედგება 3 კომპონენტისაგან (False)

2) დალაგებული ასოციაციურ კონტეინერებს იმპლემანტაციისათვის გამოიყენება თვითბალანსირებადი BST (ბინარული ძებნის ხეები) (True)

3) პოინტერები შესაძლებელია იყვნენ კლასის წევრ - ცვლადები (data members) (True)

4) დაულაგებული ასოციაციურ კონტეინერებს იმპლემანტაციისათვის გამოიყენება თვითბალანსირებადი BST (ბინარული ძებნის ხეები) (False)

5) იტერატორი იგივე თავისუფალი წვდომის (Random Access) პოინტერია(False)

6) C++ კლასის წევრ ცვლადები არიან დეფოლტად private შესაბამისად თუ მათზე წვდომა გვინდა ვიყენებთ გასაღებ სიტყვას public (True)

7) კომპაილერს არ ადარდებს სადაა განსაზვრული ტემპლპეიტ კლასის წევრ ფუნქციები: კლასის აღწერაშივე, ჰედერ ფაილში თუ გარეთ, შესაბამისი კლასის იმპლემენტაციის cpp ფაილში (False)

8) კომპაილერს არ ადარდებს სადაა განსაზვრული ტემპლპეიტ კლასის წევრ ფუნქციები: კლასის აღწერაშივე, ჰედერ ფაილში თუ გარეთ, შესაბამისი კლასის იმპლემენტაციის cpp ფაილში (False)

9) პოინტერები შესაძლებელია იყვნენ კლასის წევრ - ცვლადები (data members) (True)

10) იტერატორი იგივე თავისუფალი წვდომის (Random Access) პოინტერია (False)

11) C++ კლასის წევრ ცვლადები არიან დეფოლტად private შესაბამისად თუ მათზე წვდომა გვინდა ვიყენებთ გასაღებ სიტყვას public(True)

12) პოინტერი იტერატორია (True)

13) BST რომლის სიმაღლეა 3 შეიძლება შეიცავდეს 20 კვანძს (False)

14) BST რომლის სიმაღლეა 3 უნდა შეიცავდეს მინიმუმ 4 კვანძს(True)

15) ზედაპირული ასლი (shallow copy) დინამიურად გამოყოფს მეხსიერებას, რათა შეიქმნას იმ მონაცემების ასლები რომელთა კოპირებასაც ვახდენთ (False)

16) იტერატორი იგივე თავისუფალი წვდომის (Random Access) პოინტერია (False)

17)ზედაპირული ასლი (shallow copy) დინამიურად გამოყოფს მეხსიერებას, რათა შეიქმნას იმ მონაცემების ასლები რომელთა კოპირებასაც ვახდენთ (False)

18) როდესაც ვიყენებთ ბინარული ძებნის ალგორითმს მასივში ელემეტის მოსაძებნად, ის უნდა იყოს სორტირებული (True)

19)C++-ში დეფოლტად გვაქვს სტატიკური ბმა.დინამიური ბმისთვის საჭიროა გასაღები სიტყვა : virtual (True)

20) არ არსებობს ისეთ ცნება როგორიცაა ვირტუალური კონსტრუქტორი (True)

21) არ არსებობს ისეთ ცნება როგორიცაა ვირტუალური დესტრუქტორი (False)

22) თუ ვახდენთ კლასის წევრ ფუნქციების კლასის გარეთ განსაზღვრას,დაგვჭირდება ოპერატორი „ ; „ (False)

23)წევრ ცვლადები უნდა იყოს private რომ დავიცვათ ისინი არაწევრი ფუნქციები წვდომისაგან (True)

25) ჩამოთვალეთ ოთხი მონაცემთა სტრუქტურა რომელიც შესაძლებელია გამოყენებული იქნას აბსტრაქტული მონაცემთა ტიპის - სიის იმპლემენტაციისათვის (ADT LIST).

პასუხი : **arrays,linked list,stacks,queue**

26) ჩათვალეთ მოცემულია ტიპიური “MyList” კლასი, რომელიც იყენებს linked list -ს სიის იმპლემეტაციისათვის. მოცემულია შემდეგი დრაივერ პროგრამა ამ კლასისათვის. დაასახელეთ იმ ფუნქციების ზოგადი სახელები რომელთა გამოძახებაც ხდება სადაც კითხვის ნიშები წერია (ზოგადი სახელებია მაგალითად, კონსტრუქტორი, დესტრუქტორი, და ა.შ)

```
main()
{
   MyList L1, L2;        // -----------?  (1)
If(!(L2.isempty()))
MyList L3 = L2;   //---------------? (2)
L3.display();
L2 = L1;              //------------?  (3)
}
```

პასუხი:

კონსტრუქტორი
კოპი-კონსტრუწტორი
მინიჭების კონსტრუქტორი

27) ვთქვათ საქმე გვაქვს **დინამიური მეხსიერების გამოყოფასთან** კლასის რომელიმე **წევრ ცვლადისათვის** (dynamic memory allocation for data member). ჩამოთვალეთ ის 3 ფუნქცია რომელიც ამ შემთხვევაში კლასისათვის აუცილებლად უნდა გვქონდეს განსაზღვრული:

პასუხი

push()
pop()
empty()

28) რა იქნება შემდეგი კოდის ფრაგმენტის შედეგი? (იგულისხმეთ რომ a, b და c ცვლადების მისამართებია 0x123, 0x234, 0x245)

```
double a =0.0, b=0.0, c=25.5;
double *aptr = &a;
double *bptr = &b;
double *cptr = &c;
*cptr = 51.5;
aptr = bptr;
bptr = cptr;
*aptr = 78.67;
cout<<*aptr<<" "<<*bptr<<" "<<*cptr<<" "<<aptr<< " "<<bptr<<" "<<cptr<< " "<<endl;
```

პასუხი :

78.67 51.5 51.5 0x234 0x245 0x245

29) ვთქვათ საქმე გვაქვს **დინამიური მეხსიერების გამოყოფასთან** კლასის რომელიმე **წევრ ცვლადისათვის** (dynamic memory allocation for data member). ჩამოთვალეთ ის 3 ფუნქცია რომელიც ამ შემთხვევაში კლასისათვის აუცილებლად უნდა გვქონდეს განსაზღვრული:

პასუხი:

დესტრუქტორი
კონსტრუქტორი
მინიჭების ოპერატორი

30) დაწერეთ კოდის ფრაგმენტი, რომლითაც კომპილატორისგან მოითხოვთ ახალი მეხსიერების პორციის გამოყოფას 10 მთელი რიცხვისათვის, დაარქვით ამ მეხსიერების პორციას myRA

მინიშნება : დინამიური მეხსიერების გამოყოფაზეა საუბარი

პასუხი: int myRA = new int[10]

31) კლასის ინვარიანტობა (ვალიდაცია) იმისათვის რომ კლასის წევრი ცვლადების მნიშვნელობები კორექტულად განისაზღვროს

პასუხი: აუცილებელია

32) რისთვისაა შემდეგი ფუნქცია:

```
void fun1(node* head)
{
    if(head == NULL)
      return;
    cout<< head->data;
    fun1(head->next);
 }
```

პასუხია: სიის ელემენტების ბეჭდვისათვის

33) რას წარმოადგენს **linear probing** და **quadratic probing**?

მოიყვენეთ ზოგადი ფორმულები თითოეული მათგანისათვის

კერძოდ ეკუთვნიან პრობინგი კოლიზიის ამოხსნის სტრატეგიებს. ანუ ოუფენ ედრესინგს ეკუთვნის.

პასუხი:

linear probing წარმოადგენს კომპ.პროგრამირების პროგრამირების სქემა,რომელიც საჭიროა ე.წ ჰეშის ცხრილებში შეჯახების გადასაჭრელად.ამავდროულად იგი საჭიროა მონაცემთა სტრუქტურებისთვის.

quadratic probing კი წარმოადგენს ღია მისამართების სქემა კომპ.პროგრამირებაში ჰეშის ცხრილებში შეჯახებების გადასაჭრელად. იგი მუშაობს ორიგინალი ჰეშის ინდექსის აღებით და ამავდროულად კვადრატული მრავალკუთხედის თანმიმდევრული მნიშვნელობების დამატებით,მანამ სანამ ღია ჭრილის აღმოჩენა არ იქნება.

34) რას წარმოადგენს **linear probing** და **quadratic probing**?

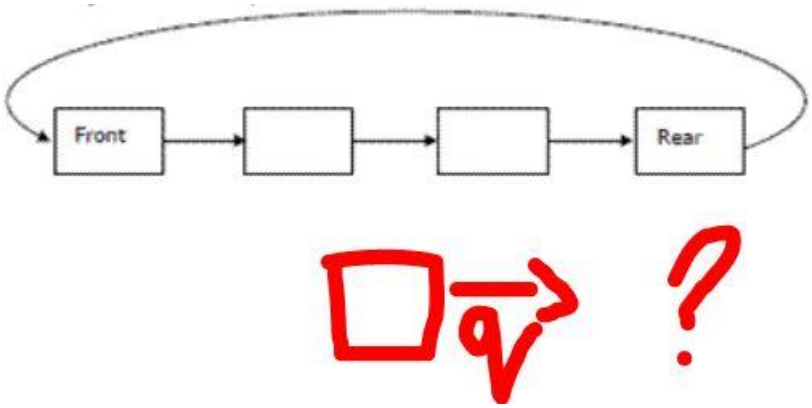
მოიყვენეთ ზოგადი ფორმულები თითოეული მათგანისათვის

პასუხი:

ხაზოვანი გამოკვლევა არის კომპიუტერული პროგრამირების სქემა ჰეშის ცხრილებში შეჯახებების გადასაჭრელად, მონაცემთა სტრუქტურებისთვის, გასაღები-მნიშვნელობის წყვილების კოლექციის შესანარჩუნებლად და მოცემულ გასაღებთან დაკავშირებული მნიშვნელობის მოსაძიებლად.

კვადრატული ზონდირება არის ღია დამისამართების სქემა კომპიუტერულ პროგრამირებაში ჰეშ-შეჯახებების გადასაჭრელად ჰეშის ცხრილებში. კვადრატული ზონდირება მუშაობს ორიგინალი ჰეშის ინდექსის აღებით და თვითნებური კვადრატული მრავალკუთხედის თანმიმდევრული მნიშვნელობებით, სანამ არ მოხდება ღია ჭრილის აღმოჩენა.

35)) წრიული ბმული სია გამოიყენება რიგის იმპლემენტაციისათვის, ცვლადი q გამოიყენება ნოდზე(კვანძზე) წვდომისათვის კლასში Queue. რომელ კვანძზე უნდა იყოს მიმთითებელი q რომ გვქონდეს მუდმივი დრო ისეთი ოპერაციებისათვის როგორიცაა enQueue და deQueue ?



პასუხია: ბოლო ელემენტზე

36) განიხილეთ შვიდ ელემენტიანი ჰეშცხრილი, საწყისი ინდექით ნული და ჰეშ-ფუნქციით (3x +2)%7. ჰეშცხრილი თავდაპირველად ცარიელია, მიუთითეთ როგორ განლაგდება მასში შემდეგი მნიშვნელობები: 1, 3, 8, 10 (კოლიზიებთან გამკლავებისათვის გამოიყენეთ linear probing.

გამოიყენეთ '_ ' სიმბოლო ცხრილში ცარიელი ადგილის აღნიშვნისთვის. მაგალითად თქვენი პასუხი უნდა გამოიყურებოდეს დაახლოებით ასე:

1, _ , 3, _ , 8, _ , 10 (7 ელემენტიან მასივში 4 ელემენტი განლაგდა შესაბამის პოზიციაზე და დარჩენილია 3 ცარიელი ადგილი)

(შენიშვნა - ეს არ არის სწორი თანმიმდევრობა - თქვენ უნდა დაწეროთ სწორი)

პასუხი: 10 _ _ _ 3 1 8

37) რა იქნება შემდეგი კოდის სირთულე (runtime complexity)? გამოიყენეთ აღნიშვნები: Θ(n), Θ(n^2), Θ(1), Θ(log(n)...)

```
for(int i=0; i<n; i++)

int count = i;

for(int j=0; j<count; j++)

for(int k=0; k<j; k++)

count--;
```

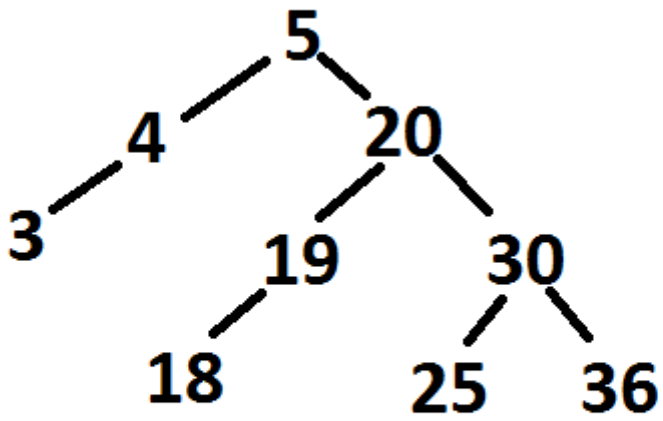
პასუხია: **Θ(n^2)**

38) მომხმარებელი შემდეგ მონაცემებს იღებს ერთმანეთის მიმდევრობით:

5, 4, 20, 30, 36, 19, 25, 3, 18

ააგეთ შესაბამისი ბინარული ძებნის ხე და ატვირტეთ ბმულ ფაილად

პასუხი:



39) რომელია empty() მეთოდის ტანის სწორი იმპლემენტა მოცემული BST-ისათვის?

```
Class BST{

class Node{

int element;

Node * left;

Node * right;

Node (int item){

        element=item;

left=0; right=0;

}

};

Node *top;

};
```

პასუხია: return top== 0;

40) სპეციალიზებული ადამიანების საძიებო ალგორითმის შემუშავების შემდეგ, ახალგაზრდა ანამ დაიწყო მისი ტესტირება. როდესაც n = 100 ადამიანის ძებნა ხორციელდებოდა 1 საათში.

შედეგების სწრაფი შემოწმების შემდეგ, მან შენიშნა, რომ ტესტი მხოლოდ მამაკების მონაცემებს მოიცავდა. მას შემდეგ, რაც მან შეცვალა ეს შეცდომა და ტესტში შეიყვანა ქალები და ბავშვები, საძიებო ნიმუშის ზომა გაიზარდა და n = 1000.

უპასუხეთ შემდეგ კითხვას.
რა დრო დაჭირდება გაზრდილი ზომის მონაცემებში ძებნას თუ ალგორითმის სირთულე არის O (n^2).

პასუხია: 100

41) თუ array[10] წარმოადგენს 10 დადებით ელემენტიან მასივს, მაშინ ქვემოთ მოცემული ფსევდო კოდი წარმოადგენს სტრატეგიას:

```
int k = 0
int m = 1
While m < 10
If a[m] < a[k] Then k = m
End If
   m = m+1
End While
```

პასუხია: მასივის უმცირესი ელემენტის მდებარეობის გარკვევისათვის

42)) რა იქნება შემდეგი კოდის სირთულე (runtime complexity) ? (გამოიყენეთ აღნიშვნები Θ(n), Θ(n^2), Θ(1), Θ(log(n))

```
int count = n;
for (int i=0; i<n; i++)
for (int j=0; j<count; j++)
for (int k=0; k<j; k++)
count--;
```

პასუხია: **Θ(n)**

43) რა იქნება შემდეგი კოდის სირთულე (runtime complexity)? გამოიყენეთ აღნიშვნები: Θ(n), Θ(n^2), Θ(1), Θ(log(n), Θ(nlog(n), ...)

```
for(int i=1; i <= n; i++)
    for(int j=1; j <= n; j*=2)
        cout<<"Hithere";
```

პასუხია: **Θ(nlog(n)**

44) მოცემულია

```
for( int row = 0; row < n; row++ ) {
        for( int col = 0; col < n; col++) {
                cout<<"Hi there";
 }
}
```

რამდენჯერ დაიბეჭდება " Hi There" თუ n = 32?

პასუხი: 1024

45) მოცემული გაქვთ ცალად ბმული სია ისეთი როგორიც კლასში განვიხილეთ. შეავსეთ გამოტოვებული ადგილები რომ მოახდინოთ ელემენტის სწორად ჩამატება (პასუხში მიუთითეთ მხოლოდ გამოტოვებულ ადგილას ჩასაწერი მნიშვნელობები მიმდევრობით 1,2,3,4,5)

```
Node *newptr = ___(1)_____  Node(data);
if (prev == 0)
{
newptr->next = __(2)_____;
  head = __(3)_____ ;
}
else
{
 newptr->next = ____(4)_____;
        ____(5)_____   =   newptr ;
}
```

პასუხი:

(1)new
(2) head
(3) newptr
(4) prev->next
(5) prev->next

46) int nigma(int value) {

```
    if( value <= 0 )
        return 1;
    return value * nigma(value-1);
}
```

რას დააბრუნებს კოდი თუ value=25 ?

პასუხია: **25!**

47) გამოიყეთ quadratic probing სტრატეგია კოლიზიებთან გამკლავებისათვის და მოიყვანეთ მიმდევრობა იმ ინდექსებისა რომლებზეც დაიჰეშება 63, თუ ჰეშირების ფუნქცია h(i)= i%20, ხოლო რიცხვების მიმდევრობაა

20, 22, 84, 42, 102, 63, 103 (ჩათვალეთ მასივის ზომაა 20)

პასუხია: **3, 4, 2, 7**

48) რამდენი კომპონენტისაგან შედგება STL ბიბლიოთეკა

პასუხი: (4)

49) ჩამოთვლილთაგან რომელი გამოიყენება მიმდევრობითი კონტეინერებისათვის სხვადასხვა ინტერფეისის მინიჭებისათვის ?

პასუხი: **კონტეინერის ადაპტერები**

50)მაქსიმალური რაოდენობა კვანძებისა სრულ (full) ბინარულ ხეში, რომლის სიღრმე არის k ?

დაწერეთ ზოგადი ფორმულა k-ს გამოყენებით იგულისხმეთ რომ k>=1

პასუხი: **2^(k+1)-1**

51) Postorder შემოვლა ბინარული ძების ხეში :

პასუხია:

რეკურსიულად მოივლის მარცხენა ქვეხეს, შემდგომ მარჯვენა კვეხეს და ეწვევა კვანძს (root)

52)რომელია ელემენტის ძებნის უარესი (worst case) და საშუალო (average case) შემთხვევის ეფექტურობა ბინარული ძების ხისათვის ?

პასუხია: **O(n), O(log(n))**

53)ჰეშირების ობიექტური მიზნებია ელემენტის ჩამატება და ძებნა განახორციელოს O(1) , თუმცა ამავდროულად

პასუხია: **მოითხოვება შესანახი სივრცის მინიმიზაცია**

54) დაასრულეთ წინადადება:

ობიექტის ჰეშკოდი _____

პასუხია:

არის მთელი რიცხვი, რომლის საშუალებითაც შესაძლებელია ობიექტის დახასიათება და მისი იდენტიფიცირება

55) როდესაც ვიყენებთ ბინარული ძებნის ალგორითმს მასივში ელემეტის მოსაძებნად, ის უნდა იყოს სორტირებული (True)

56) ჩამოთვლილთაგან რომელი გამოიყენება მიმდევრობითი კონტეინერებისათვის სხვადასხვა ინტერფეისის მინიჭებისათვის ?

პასუხი: **კონტეინერის ადაპტერები**

57) გამოიყეეთ quadratic probing სტრატეგია კოლიზიებთან გამკლავებისათვის და მოიყვანეთ მიმდევრობა იმ ინდექსებისა რომლებზეც დაიჰეშება 63, თუ ჰეშირების ფუნქცია h(i)= i%20, ხოლო რიცხვების მიმდევრობაა

პასუხი: **3, 4, 2, 7**

58) რომელია ელემენტის ძებნის უარესი (worst case) და საშუალო (average case) შემთხვევის ეფექტურობა ბინარული ძების ხისათვის ?
პასუხი: **O(n), O(log(n))**

59) თუ array[10] წარმოადგენს 10 დადებით ელემენტიან მასივს, მაშინ ქვემოთ მოცემული ფსევდო კოდი წარმოადგენს სტრატეგიას:

```
int k = 0

int m = 1

While m < 10

If a[m] < a[k] Then k = m
End If
   m = m+1
End While
```

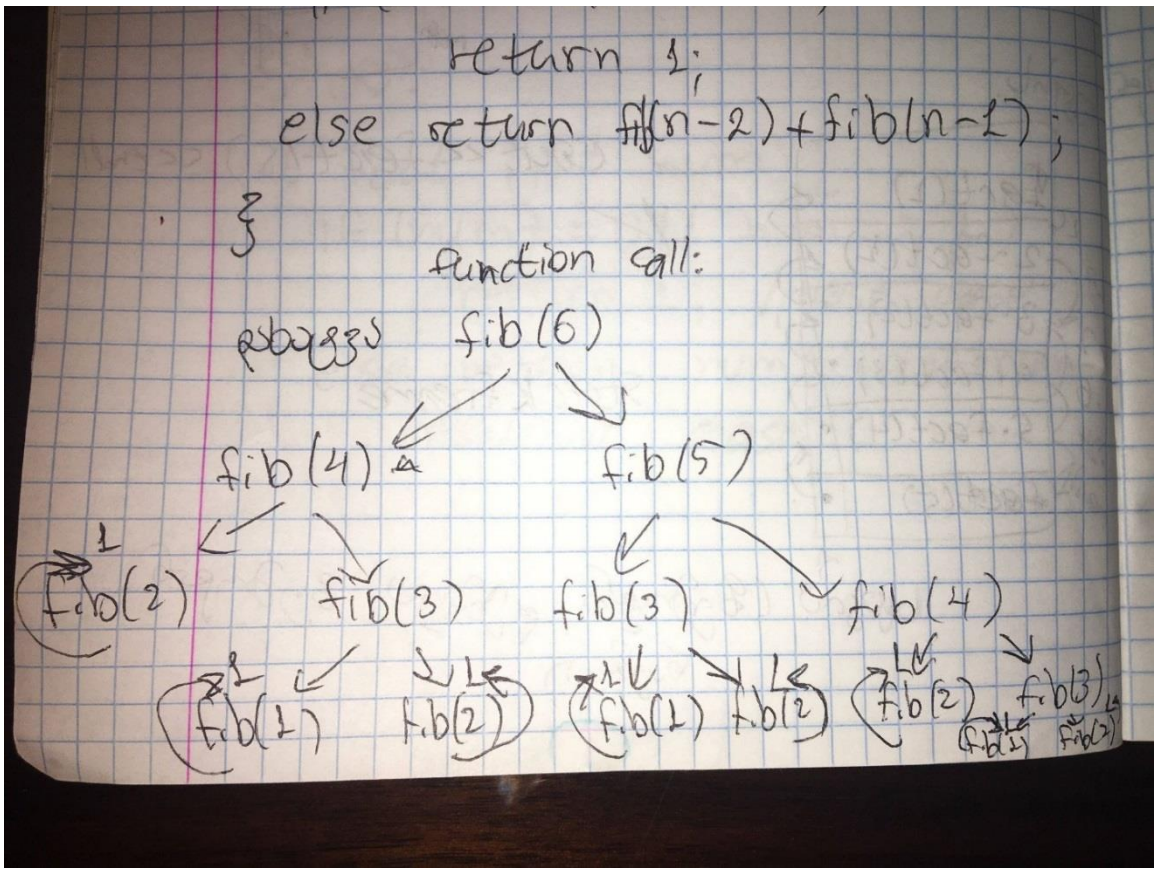
პასუხი: **მასივის უმცირესი ელემენტის მდებარეობის გარკვევისათვის**

60) ჰეშირების ობიექტური მიზნებია ელემენტის ჩამატება და ძებნა განახორციელოს O(1) , თუმცა ამავდროულად

პასუხი: **მოითხოვება შესანახი სივრცის მინიმიზაცია**

61) ვემოთ მოცემული ფუნქციისათვის, დახატეთ ფუნქციათა გამოძახების ჯაჭვი თუ x = 6

```
int fib(int x){
   if((x==1)||(x==0)){
      return(x);
   }else{
      return(fib(x-1)+fib(x-2));
   }}
```



62) კოლიზიებთან გამკლავების რომელი სტრატეგიები იცით? რომელი როდის არის უმჯობესი და რატომ?

პასუხი:

ღია და დახურული ედრესინგი ღიაში გვაქვს წრფივი და კვადრატული პრობინგი და დახურულში ჩენინგი იმისთვის რომ კლასტერები არ შეგვექმნას კარგია ჩენინგის გამოყენება.

63) მოცემული გაქვთ შემდეგი ბინარული ძებნის ხე, დაწერეთ მისთვის კვანძის ჩამატების მეთოდი

```
Class BST {
        class Node {
        int element;
        Node * left;
        Node * right;
        Node (int item) {
              element=item;
               left=0; right=0;
        }
    };
    Node *top;
};
```

პასუხი :

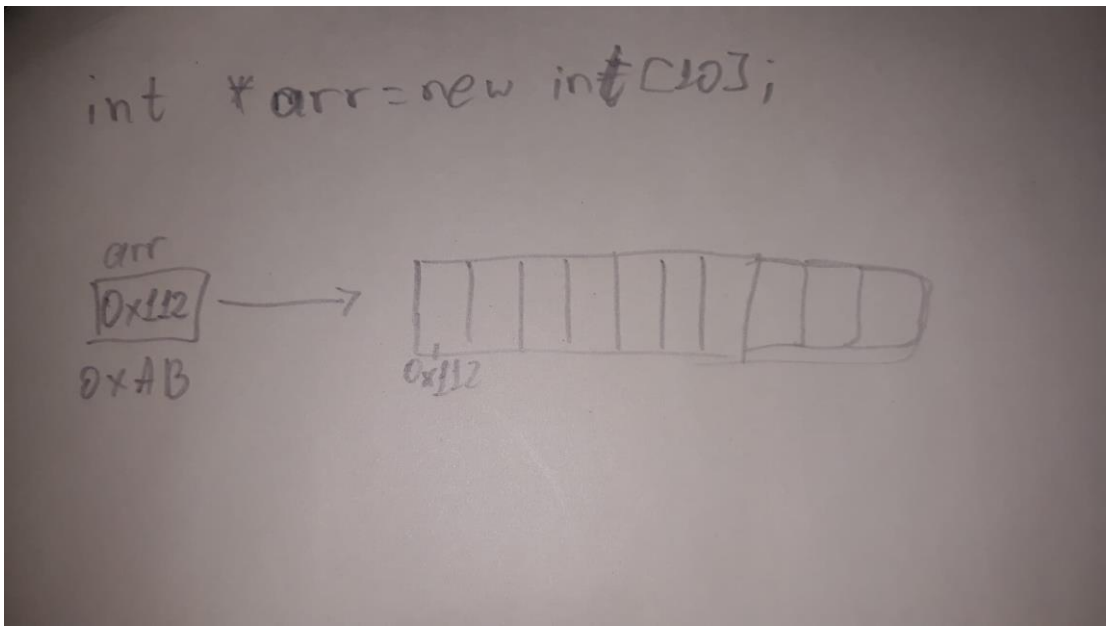
```
Class BST {

        class Node {

        int element;

        Node * left;

        Node * right;
```

```
        Node (int item) {

            element=item;

             left=0; right=0;

        }

    };

    Node *top;

};

void insert(int add){

Node *thisEl=top;

Node *par=0;

bool found=false;

if(top== 0)

            thisEl=new Node(add)
```

```
else{

    while(!found && thisEl!=0){

    par=thisEl;

    if(add>thisEl->element)

            thisEl=thisEl->right;

    else if(add<thisEl->element)

            thisEl=thisEl->left;

    else found=true;

}
```

64) დახატეთ დიაგრამა, რომელზეც ასახავთ 10 სიმბოლო (char) ტიპის მნიშვნელობების განსათავსებლად შექმნილი დინამიური მასივის მეხსიერების გამოყოფას. ჩათვალეთ რომ პირველი ბაიტის მისამართია 0xAB. პოინტერს, რომელიც ამ მეხსიერებაზე უთითებს აქვს მისამართი 0x123

პასუხი:



65) დახატეთ მეხსიერების გამოყოფის დიაგრამა, რომ აჩვენოთ მეხსიერების განაწილება შემდეგი ცვლადებისათვის:

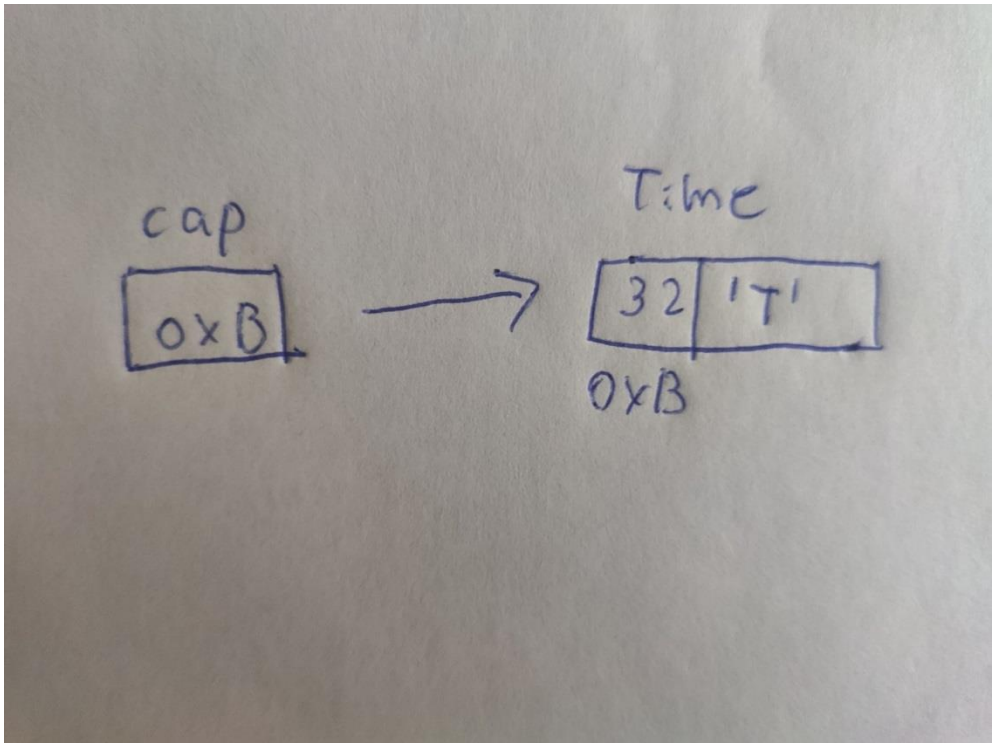
class Dog { int age; int price; char Initial;};

Dog *Dptr = new (nothrow) Dog;

Dptr->age= 5;

Dptr->price = 200;

Dptr->Initial = 'B';



66) მოცემული გაქვთ შემდეგი კლასი, უპასუხეთ ქვემოთ მოცემულ კითხვებს?

```
class My_class{
public:
   My_class();
   My_class(char c, int i, float f);
   int get_ID();
   My_class &  set_char(char m);
   void display(ostream& p) const;
private:
   char name;
   int id;
   float gpa;
}
void operator<<(ostream & out, My_class & C){
//შესაბამისი კოდი
}
```

ახსენით რა მოხდება შემდეგი ორი კოდის ფრაგმენტის გაშვებისას (იმსჯელეთ თითოეულზე ცალ-ცალკე):

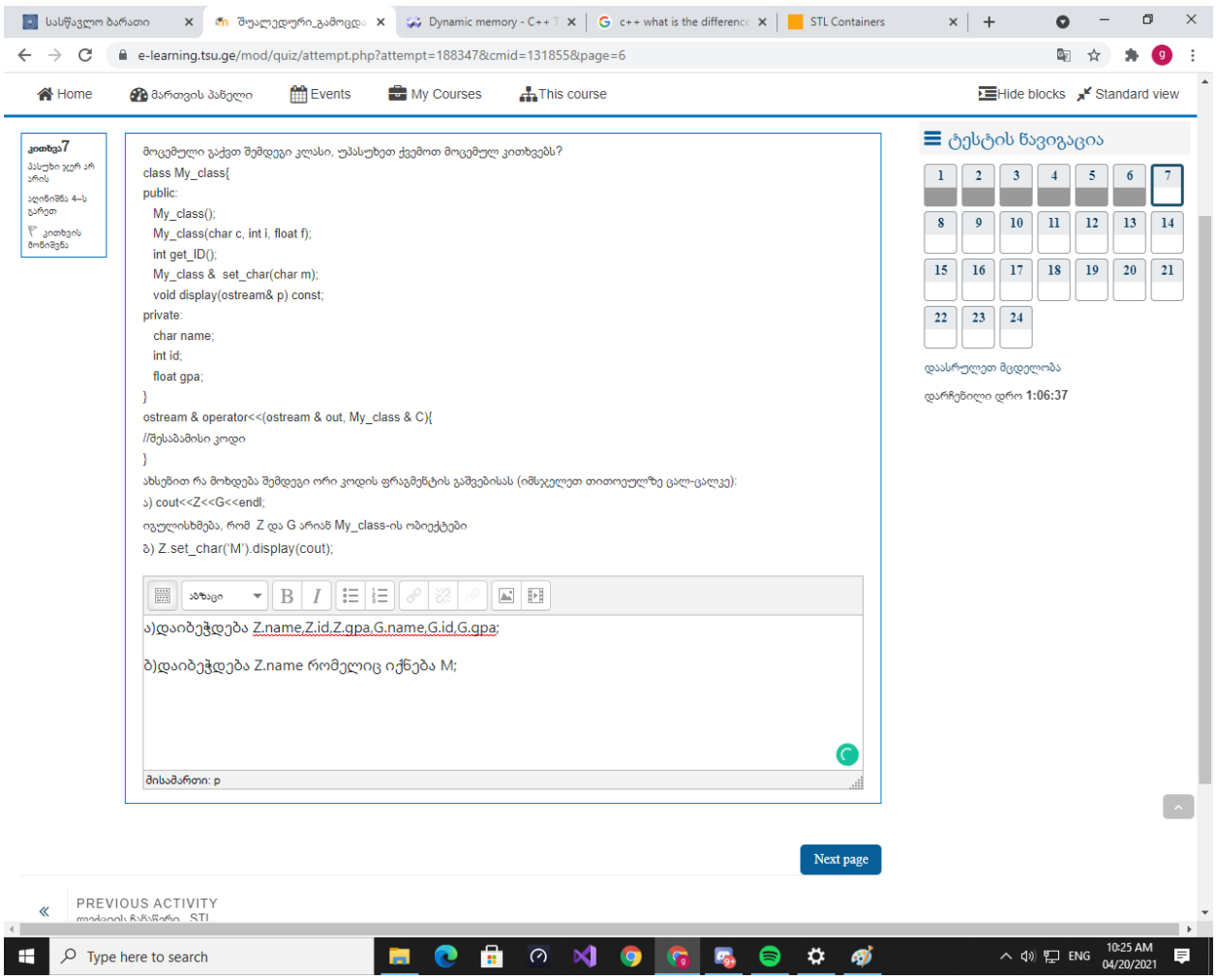
ა) cout<<Z;

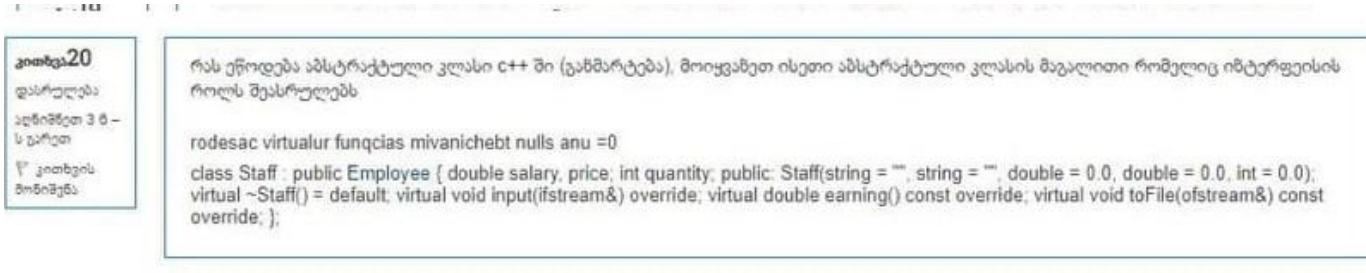
ბ) cout<<G<<endl;

იგულისხმება, რომ Z და G არიან My_class-ის ობიექტები

გ) Z.set_char('M').display(cout);

პასუხი : ა) გაშვებისას დაიბეჭდება Z ის და G ს დისფლეის იმპლემენტირებული ანუ დაიბეჭდება ID GPA და Name ბ) შეცვლის name ველის მნიშვნელობას და დაბეჭდავს


e-learning.tsu.ge/mod/quiz/attempt.php?attempt=188347&cmid=131855&page=6
Home
Events
My Courses
This course
Hide blocks
Standard view
კითხვა 7
პასუხი ჯერ არ არის
აღნიშნვა 4-ს გარეთ
კითხვის მონიშვნა
მოცემული გაქვთ შემდეგი კლასი, უპასუხეთ ქვემოთ მოცემულ კითხვებს?
class My_class{
public:
My_class();
My_class(char c, int i, float f);
int get_ID();
My_class & set_char(char m);
void display(ostream& p) const;
private:
char name;
int id;
float gpa;
}
ostream & operator<<(ostream & out, My_class & C){
//შესაბამისი კოდი
}
ახსენით რა მოხდება შემდეგი ორი კოდის ფრაგმენტის გაშვებისას (იმსჯელეთ თითოეულზე ცალ-ცალკე):
ა) cout<<Z<<G<<endl;
იგულისხმება, რომ Z და G არიან My_class-ის ობიექტები
ბ) Z.set_char('M').display(cout);
აზზაცი
ა)დაიბეჭდება Z.name,Z.id,Z.gpa,G.name,G.id,G.gpa;
ბ)დაიბეჭდება Z.name რომელიც იქნება M;
მისამართი: p
ტესტის წავიგაცია
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24
დაასრულეთ მცდელობა
დარჩენილი დრო 1:06:37
Next page
PREVIOUS ACTIVITY
Type here to search
ENG
10:25 AM
04/20/2021


კითხვა 20
დასრულება
აღნიშნეთ 3 ტ – ს გარეთ
კითხვის მონიშვნა
რას ეწოდება აბსტრაქტული კლასი c++ ში (განმარტება). მოიყვანეთ ისეთი აბსტრაქტული კლასის მაგალითი რომელიც იმპერფეისის როლს შეასრულებს
rodesac virtualur funqcias mivanichebt nulls anu =0
class Staff : public Employee { double salary, price; int quantity; public: Staff(string = "", string = "", double = 0.0, double = 0.0, int = 0.0); virtual ~Staff() = default; virtual void input(ifstream&) override; virtual double earning() const override; virtual void toFile(ofstream&) const override; };

დაწერეთ ფუნქცია რომელიც წაშლის მონაცემად მიღებული ცალად ბმული სიის კვანძს, კვანძ კლასს აქვს შემდეგი სახე:

```
class ListNode {
  public:
   int val;
   ListNode *next;
   ListNode(int x) : val(x), next(NULL) {}
};

void deleteNode(ListNode* node) {
   // თქვენი კოდი დაწერეთ აქ;
}
```

აბზაცი

```
node->val=node->next->val;

node->next=node->next->next;
```

კითხვა 17
დასრულება
აღნიშნეთ 1 4 – ს გარეთ
კითხვის მონიშვნა

float -ების მასივისათვის რომლის სახელია RA და 7 ელემენტს შეიცავს დაწერეთ კოდი, რომელიც მოახდენს ელემენტების წაძვრას [i+ იწდექსიდან [i]-ზე , ხოლო ბოლო ელემენტს გამუსაზღვრეთ მნიშვნელობა 7.7

```
float RA[ 7 ]={0.0};
for(int i=0; i<5; i++){
        RA[i+1]=[i];}
RA[6]=7.7;
```

კომენტარი:
ukugma unda ikos da gakliat masivis saxeli, ratom atrialebt cikls 5-mde?
`RA[i]=RA[i+1]`

კითხვა 18
დასრულება
აღნიშნეთ 3 4 – ს გარეთ
კითხვის მონიშვნა

რამდენნაირი იტერატორი გვაქვს - ჩამოთვალეთ და მოითითეთ ყველაზე "სრულყოფილი" (ფუნქციონალის თვალსაზრისით) მათ შორის

random-
access,brdirectional,forward,output,intp

კომენტარი:

1) ჩამოთვალეთ სამი მონაცემთა სტრუქტურა,რომელიც შესაძლებელია გამოყენებული იქნას აბსტრაქტული მონაცემთა ტიპის -სიის იმპლემენტაციისათვის

პასუხი:
სტატიკური მასივი
დინამიური მასივი
ბმული სია

2) მოიყვანეთ ტემპლეიტ ფუნქციის მაგალითი,რომელიც ახდენს მაქსიმალური მნიშვნელობის დაბრუნებას

პასუხი:

```
template<typename T>
T& returnGreater(T & A, T&B) {
if(A>B)
{Return A; }
else { return B;}}
```

3) ჩამოთვალეთ -ის ძირითადი კომპონენტები

პასუხი:
1.ალგორითმები
2.კონტეინერები
3.ფუნქციები
4.იტერატორები

4) რომელი მემკვიდრეობითობის ტიპი არ არსებობს c++ ში

პასუხი: **ასიმპტოტური**

დასკვნითი გამოცდის საკითხები

1)შეგვიძლია ვთქვათ რომ Merge Sort უფრო სწრაფია ვიდრე Insertion Sort მთლიანად დაუსორტავი საკმაო ზომის მონაცემების სორტირებისას

აირჩიეთ ერთი:

True

2) C ++ იძლევა საშუალებას კომპილატორს მიუთითოს რომელი ფუნქცია უნდა მიუსადაგოს ობიექტს კომპილაციის დროს. ამ პროცესს ეწოდება სტატიკური ბმა

3) Postorder შემოვლა ბინარული ძების ხეში :
რეკურსიულად მოივლის მარცხენა ქვეხეს, შემდგომ მარჯვენა კვეხეს და ეწვევა კვანძს (root)

4) რა იქნება შემდეგი კოდის შედეგი და რატომ? (სინტაქსური შეცდომები არ გვაქვს ჩათვალეთ...

```
class Base

{

public:

   void show() { cout<<" In Base \n"; }

}
```

```
class Derived: public Base
```

{

public:

void show() { cout<<"In Derived \n"; }

}

გამოძახება

Base *bp = new Derived

bp->show() _____(1)

Base &br = *bp

br.show() _____ (2)

in base

in base

4) ა იქნება შედეგი migrate() -ის გამოძახებისას შემდეგ კოდის ფრაგმენტში? (იგულისხმეთ რომ სინტაქსური შეცდომები არ გვაქვს და რაც საჭიროა იმპლემენტირებულია)

class bird

```
{

public:

 virtual void fly() = 0;

 void migrate(bird* tweetiepie)

 {

 // ...

 tweetiepie->fly();

 // ...

 }




};

class swallow : public bird

{

public:
```

```
  virtual void fly()

  {

    cout<<"flap_wings_like_crazy";

  }

};



class albatross : public bird

{
public:
  void fly()
    cout<<"glide_majestically_over_the_waves";
  }
};

int main(){
swallow s;
albatross a;
s.migrate(&a); // _____Answer 1
a.migrate(&s);    //_____Answer 2
return 0;}
```

answer 1) _"glide_majestically_over_the_waves"

answer2) "flap_wings_like_crazy"

5) განიხილეთ შემთხვევა როდესაც swap ოპერაცია ძალიან "მძიმეა"(ბევრი რესურსი მიაქვს). შემდეგი ალგარითმებიდან რომელს უნდა მიანიჭოთ უპირატესობა რომ მოახდინოთ swap -ების რაოდენობის მინიმიზაცია?

Selection Sort

6) კონსტრუქტორი შეგვიძლია გადავტვირთოთ True

7) ჩამოთვლილთაგან რომელია ჭეშმარიტი რიგის ბმული სიით იმლემენტაციის შემთხვევაში? ორივე ზემოაღნიშნული

8) შეგვიძლია ვთქვათ რომ Bubble sort (მოდიფიცირებული) უფრო სწრაფია ვიდრე HeapSort სრულად დაულაგებელი საშუალო ზომის მონაცემების სორტირებისას False

9) რა საერთო და განმასხვავებელი მახასიათებლები აქვთ AVL და Red Black ხეებს?

AVL უფრო სწრაფია,ვიდრე Red Black

Red Black კი უფრო სწრაფია ჩამატებაში და წაშლაში.მას სჭირდება ერთი ბიტი ერთ ნოდზე ,AVL-ს კი მეტი უნდა რადგან ინახავს სიმაღლის მნიშვნელობას.ყოველ ნოდზე საბაზისო ოპერაციებისთვის ორივეს სჭირდება O(logn) დრო

10) Quicksort ალგორითმის worst case არის O(n^2)

True

11) Quick Sort უნდა იყოს უკანასკნელი არჩევანში თუ მასივი რომლის სორტირებასაც ვახდენთ უკვე სორტირებული ან თითქმის სორტირებულია True

11) რომელია worst case დროითი სირთულე (time complexity) ძებნის, ჩამატებისა და წაშლის ოპერაციებისათვის ჩვეულებრივი ბინარული ძებნის ხისათვის (BST)? O(პასუხი n

12) რამდენი სახის მემკვიდრეობა არსებობს? ჩამოთვალეთ ისინი?

6 სახის მემკვიდრეობა არსებობს..ისინია

1) Multi-level inheritance

2)Single inheritance

3)Multie inheritance

4)Hybrid interitance

5)Multipath interitance

6)Hierarchical inheritance

13) წაშლის ოპერაცია BST ში ხორციელდება მარცხენა ქვეხის უკიდურესი მარცხენა ელემენტის გამოყენებით თუ წასაშლელ კვანძს ყავს 2 შვილი

False

14) რომელია best და worst შემთხვევის ეფექტიანობა ბინარული ძებნის ხისათვის? best - O(log(n)) worst - O(n)

15) მოიყვანეთ მარჯვენა მობრუნების კოდი (მარტო ტრიალი) შემდეგი AVL ხისათვის რომლის კვანძს აქვს სახე:

```
class Node
{
   public:
      int key;
      Node *left;
      Node *right;
      int height;
```

```
};
void  *rotRight(Node *y)
{
//თქვენი კოდი აქ
}
y* L=root->1
y*Y=root->r;
L->r=root;

root->1=Y;

root=L;
```

16) მეგობარ (friend) ფუნქციას არ ჭირდება public წევრი ფუნქციების დახმარება private წევრებზე წვდომისათვის

True

17) თუ h არის ნებისმიერი ფუნქცია და გამოიყენება n გასაღებების(key) ჰეშირებისათვის ცხრილში რომლის ზომაა m, სადაც n <= m,

შეჯახების(collision) სავარაუდო რაოდენობა კონკრეტული გასაღებისათვის ნაკლებია 1-ზე False

18) best case სცენარი წრფივი insertion sort -ისათვის არის O(n)

აირჩიეთ ერთი:

True

19) რა არის სორტირების საუკეთესო დროითი სირთულე (complexity) ? O(____) სწორი პასუხია: n

20) მასივში ელემენტის ძებნისათვის, ჩამოთვლილი ალგორითმებიდან რომელი წარმოადგენს საუკეთესოს, მისი ტიპიური იმპლემენტაციის

შემთხვევაში თუ ეს მასივი არის სორტირებული ან თითქმის სორტირებული? Insertion Sort

21) MargeSort ალგორითმის worst case არის O(n^2) False

22) C++ მემკვიდრე კლასმა რომ მოახდინოს მშობელი კლასისაგან მიღებული ფუნქციის გადაფარვა (override) ეს ფუნქცია შვილ კლასში უნდა გამოვაცხადოთ როგორც ვირტუალური virtual False

23) insertion sort ალგორითმის მუშაობის პრინციპი ეფუძნება შემდეგს: მასივის დაულაგებელი პორციიდან იღებს ელემენტს და ათავსებს მას შესაბამის ადგილას მასივის უკვე დალაგებულ პორციაში

24) 4 ელემენტიანი მასივის ზრდადობით ვალაგებთ insertion sort ალგორითმის გამოყენებით. რამდენ შედარებას განახორციელებს ეს ალგორითმი თუ მასივი თავიდან კლებადობით იყო დალაგებული?
6

25) Selection Sort უნდა იყოს ჩვენი არჩევანი თუ მასივი უკვე სორტირებულია False

26) Inline ფუნქციათა გამოძახება რანთაიმის დროს ხდება 'False'.

27) ჰეშ ცხრილში ძებნის სასურველი დროა O(___) 1

28) worst case სცენარი წრფივი insertion sort -ისთვის არის O(n^2)

აირჩიეთ ერთი:

True

29) შეგვიძლია ვთქვათ რომ QuickSort არის უფრო სწრაფი ვიდრე Insertion Sort თითქმის სორტირებული მონაცემების სორტირებისათვის.
False'.

30) შეგვიძლია ვთქვათ რომ Selection sort (სტანდარტული) უფრო სწრაფია

ვიდრე Bubble sort (სტანდარტული) საშუალო ზომის მონაცემების სორტირებისათვის

აირჩიეთ ერთი:

True

31) თუ array[10] წარმოადგენს 10 დადებით ელემენტიან მასივს, მაშინ ქვემოთ მოცემული ფსევდო კოდი წარმოადგენს სტრატეგიას:

```
int k = 0

int m = 1

While m < 10

If a[m] < a[k] Then k = m
End If
   m = m+1
End While
```

c. მასივის უმცირესი ელემენტის მდებარეობის გარკვევისათვის

29) მშობელი კლასი გვთავაზობს განზოგადებულ ვერსიას შვილი კლასისათვის

აირჩიეთ ერთი:

True

30) როდესაც ხდება მშობელი კლასის წევრების public წესით

მემკვიდრეობით მიღება, ისინი შვილ კლასში private ნაწილში განთავსდებიან False

31) ჩამოთვლილი ალგორითმებიდან რომელი არ არის სტაბილური მათი ტიპიური იმპლემენტაციის შემთხვევაში?

აირჩიეთ ერთი:

a. Quick Sort Co

32) ქვემოთ ჩამოთვლილთაგან რომელი წარმოადგენს სიმართლეს სტეკის ბმული სიით იმპლემენტაციის შემთხვევაში?

აირჩიეთ ერთი:

a.

დამატების ოპერაციის შემთხვევაში, თუ ახალი კვანძი ემატება ბმული სიის დასაწყისში, მაშინ წაშლის ოპერაციის შემთხვევაში,

კვანძები უნდა ამოიღონ ბოლოდან

b.

დამატების ოპერაციის შემთხვევაში, თუ ახალი კვანძი ემატება ბმული სიის ბოლოში, მაშინ წაშლის ოპერაციის შემთხვევაში,

კვანძები უნდა ამოიღონ თავიდან

c. ორივე

d. არცერთი Correct

30)