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

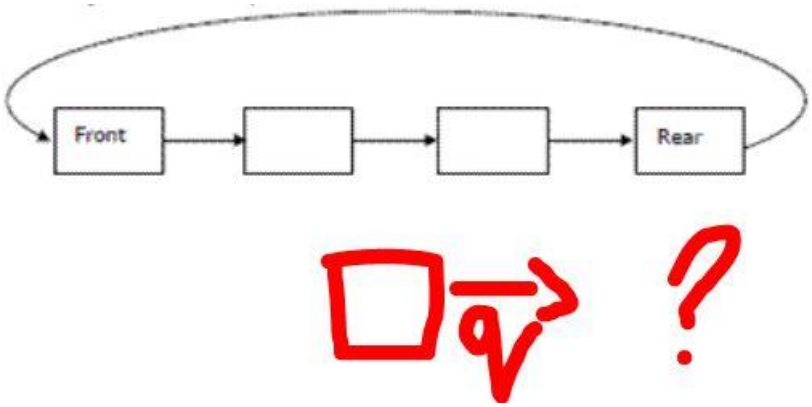
კერძოდ ეკუთვნიან პრობინგი კოლიზიის ამოხსნის სტრატეგიებს. ანუ ოუფენ ედრესინგს ეკუთვნის.

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.

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--;
```

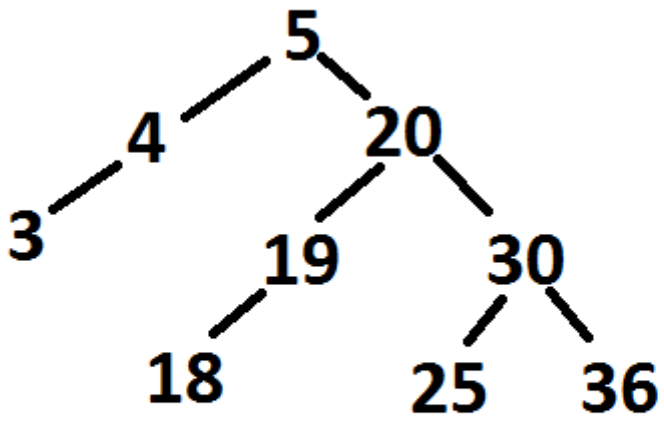
პასუხია: **Θ(n^2)**

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) გამოიყეეთ 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**

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

პასუხი:

სტატიკური მასივი
დინამიური მასივი
ბმული სია

3) ჩამოთვალეთ -ის ძირითადი კომპონენტები

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

4) რომელი მემკვიდრეობითობის ტიპი არ არსებობს c++ ში

პასუხი: ასიმპტოტური