#Page 50-Exercises

```cpp
bool bs(int st, int end , vector<int>v, int x)
{
    while(st<=end)
    {
        int mid = st + ( end - st) / 2;
        if(v[mid]==x) return true;
        else if(v[mid] > x) end = mid - 1;
        else st = mid + 1;
    }
    return false;
}

int main()
{
    vector<int>v = {20,44,48,55,62,66,74,88,93,99};
    cout << bs(0,9,v,44) << '\n';
}
```

#Page 50-Exercises
#(1)

#using frequency array method to solve in O(n).

```cpp
int main()
{
    vector<int>lst={1,1,2,2,2,4,4,5},freq(200005);
    for(int i=0 ; i<lst.size() ; i++)
    {
        freq[lst[i]]++;
    }
    int mx = 0, res = 0;
    for(int i=0 ; i<lst.size() ; i++)
```

```
    {
        if(freq[lst[i]]>mx)
        {
            mx = freq[lst[i]];
            res = lst[i];
        }
    }
    cout << res << "\n";
}
```

```
#(2)
int main()
{
    vector<int>lst={1,1,1,1,2,2,2,2},freq(200005);
    for(int i=0 ; i<lst.size() ; i++)
    {
        freq[lst[i]]++;
    }

    int res=0;
    for(int i=0 ; i<lst.size() ; i++)
    {
        if(freq[lst[i]]>=(lst.size()/2))
        {
            res = max(res, lst[i]);
        }
    }
    cout << res << '\n';
}
```

```
#(3)
#using binarySearch, can be solved in O(n*log(m))
instead of O(n*m) where n=len(x) and m=len(y).
```

```cpp
int main()
{
    vector<int>x={1,2,3},y={1,2,3};
    int val=2;

    for(int i=0 ; i<x.size(); i++)
    {
        int res = bs(0, y.size()-1, y, val-x[i]);
        if(res != -1)
        {
            cout << x[i] << " " << res << '\n';
            return 0;
        }
    }
    cout << "Not found\n";
}
```

 # Also can be done in O(n+m) using hashmap.

```cpp
int main()
{
    vector<int>x={2,2,3},y={1,2,3};
    int val=3;

    map<int,int>mp;
    for(int i=0 ; i<x.size() ; i++)
    {
        mp[x[i]]=val-x[i];
    }

    map<int,bool>IsInY;
    for(int i=0 ; i<y.size() ; i++)
    {
        IsInY[y[i]] = true;
    }
```

```
    for(int i=0 ; i<x.size() ; i++)
    {
        int res = mp[x[i]];
        if(IsInY[res])
        {
            cout << x[i] << " " << res << '\n';
            return 0;
        }
    }
    cout << "Not found\n";
}
```

#(4)
#using frequency array method to solve in O(n).

```
int main()
{
     vector<int> lst = {1, 1, 2, 2, 2, 4, 4,
5},freq(200005);

    for (int i = 0; i < lst.size(); i++)
    {
        freq[lst[i]]++;
    }
    for(int i=0 ; i<lst.size() ; i++)
    {
        cout << freq[lst[i]] << " ";
    }
```

```
#(5)
#using two-pointers method, can be solved in O(n^2)
instead of O(n^3).

int main()
{
    vector<int> lst = {1, 1, 3, 2, 1, 4, 4, 5};
    int val = 8;
    sort(lst.begin(), lst.end());

    for(int i=0 ; i<lst.size()-2 ; i++)
    {
        int l=i+1, r=lst.size()-1;
        while(l<r)
        {
            int sum = lst[i]+lst[l]+lst[r];
            if(sum==val)
            {
                cout << lst[i] << " " << lst[l] << "
" << lst[r];
                return 0;
             }
             else if(sum<val) l++;
             else r--;
         }
    }
    cout << "Not found\n";
}
```

#(6)
#return insert position 1-based.

```
int bs(int st, int end , vector<int>v, int x)
{
    int mid;
    while(st<=end)
    {
        mid = st + ( end - st) / 2;
        if(v[mid]==x) return mid+1;
        else if(v[mid] > x) end = mid - 1;
        else st = mid + 1;
    }
     return end+2;
}
int main()
{
 vector<int> lst = {2,3,7,8};
 cout << bs(0,3,lst,1) << '\n'; }
```

#Page 105-Exerises.

#(1)

```
int main()
{
     double n; cin >> n;
     const double pi = 3.14;
     cout << "Diameter: " << n*2 << '\n';
     cout << "circumference: " << (2*n)*pi << '\n';
     cout << "surface area: " << 4*pi*(n*n) << '\n';
     cout << "volume: " << (4/3)*pi*(n*n*n) << '\n';
}
```

#(2)

```
int main()
{
    int n; cin >> n;
    double res = 0;
    double d=1, sign=1;

    while(n--)
    {
        res+= (1/d)*sign;
        d+=2;
        sign*=-1;
    }
    cout << res << '\n';
}
```

#(3)

```
int main()
{
    int n; cin >> n;
    while(n>=0)
    {
        cout << n << '\n';
        n--;
    }
}
```

#(4)
#I just learned Binet's Formula and couldn't wait to use it.

```
int main()
{
    double sum = 0, i=1, fib = 0;
    while (true)
    {
        fib = (pow(((1+sqrt(5))/2),i)-pow(((1-
        sqrt(5))/2),i))/sqrt(5);

        if(fib>500) break;
        if((int)fib%2==0) sum+=fib;
        i++;
    }
    cout << sum << '\n';
}
```

#Page 191

#(1)

```
queue<int>q;
q.push(1); q.push(2); q.push(3);
queue<int>ans;

while(!q.empty())
{
    ans.push(q.front()); ans.push(q.front());
    q.pop();
}
```

```
#(2)
int main()
{
    queue<string>q,ans;
    q.push("a"); q.push("b"); q.push("c");

    stack<string>st;
    while(!q.empty())
    {
       ans.push(q.front());
       st.push(q.front());
       q.pop();
    }
    while(!st.empty())
    {
       ans.push(st.top()); st.pop();
    }
}
```

```
#(3)
def evalPostfix(exp):
    stack = Stack()
    for i in exp:
        if i>='0' and i<='9':
            stack.push(i)
        else:
            if stack.getSize() < 2:
                return "Wrong expression!"
            x=stack.pop()
            y=stack.pop()
            stack.push(str(eval(x+i+y)))

    return float(stack.pop())
```

```
#Page 225
class Node(object):
    def __init__(self, data=None, next=None):
        self.data = data
        self.next = next

class linkedList(object):
    def __init__(self):
        self.root = None
        self.tail = None
        self.size = 0

    def addFront(self, data):
        newNode = Node(data)
        if not self.size:
            self.root = self.tail = newNode
            newNode.next = None
        else:
            newNode.next = self.root
            self.root = newNode
        self.size+=1

 def addLast(self, data):
        newNode = Node(data)
        if not self.size:
            self.root=self.tail=newNode
            newNode.next = None
        else:
            self.tail.next = newNode
            self.tail = newNode
            newNode.next = None
        self.size+=1
 def insert(self, data, pos):
        if pos<=0 or pos>self.size+1:
```

```
            return print("Invalid position")
        newNode = Node(data)
        current = Node()
        if(pos==1):
            self.addFront(data)
        elif(pos==self.size+1):
            self.addLast(data)
        else:
            current = self.root
            for i in range(1,pos-1):
                current = current.next
            newNode.next = current.next
            current.next = newNode
        self.size+=1

 def printData(self):
        current = Node()
        current = self.root
        while current is not None:
            print(current.data)
            current = current.next

llst = linkedList()
arr = [1,2,3,4,5]
for i in arr:
    llst.addLast(i)
```

----------MCQ Q's----------

#Page 245–

1-Constant time 2-Nodes 3-Linear time 4-At the end
5-Next and previous

#Page 38–

1) a-2^n b-2^n c-n^3
2) A does more work, for n=1, both are the same, for n>1, A=2B-n
3) O(n^2)
4) O(1)
5) O(n*sqrt(n))

#Page 138–

b-b-b-b-b-a-b-a-b-a

#-Q's I have no idea where they are but my friends told about----------

1-A binary search assumes that the data are:

a) Arranged in no particular order

b) Sorted (T)

2- A selection sort makes at most:

a) n^2 exchanges of data items (T)

b) n exchanges of data items

3- An example of an algorithm whose best-case, average-case, and worst-case behaviors are the same is:

a) Sequential search

b) Selection sort (T)

c) Quicksort

4. The recursive Fibonacci function makes approximately:

b) 2^n recursive calls for problems of a large size n (T)

أنس ماهر أحمد الهريجي
804615907
data structures