

bend Min (dibb): def index = -1 min = 99999999 for i in range (den (dibb)): if (not diff[i][i] and minimones dibb [i] [o] : min= diff [i][o] index = i xelui neuter det shortest Seek Time First (request, head): if (len (request) = = 0): return d= len (venuest) diff = 10] * 1 for V in range (l): diff [i] = [0, 0] Sell_count = 0 Sell- Sequence = [0] * (1+1)

→

for i in range (l): Seek-Seavence [i] = head calculate Bibb (request, nead, dibb) index = find Min (dibb) dilb [index][1] = True Seek_count += diff [index][0]. head = request [index] Seek-sequence [den Cleck-sequence) - 1] z head print l'Total number of seek operations=", seek_ count) print (" Seek Seavence is") for i in range (IH): print (leek - Seavence [i]) 4 _ name _ = _ / _ / main __ ": proc = C/46,79, 34, 60, 92, 11,41, 114] Shortest Seek Time First (proc, 50)

Scanned by CamScanner

```
Q_3
      C++ code for SCAN disk 8 cheduling
      algorithm
     # include < bets / stdc++ h>
A.
     using namespace std;
     int size = 8;
     int disk_Size = 200;
     void SCAN (int aral], int head, string direction)
         int seek_count =0;
         unt distance, un tracki
         vector < int > left, right,
         vector < int > Seen Sequence;
         if (direction = = left')
           lest. push-back (0);
         else if ( direction = = 'right')
             right/push_back (disk_Size -1)
         for (int i=0; i < size; i++) &
             y (arrij < head)
                left. push_back (arrCi]);
             4 ( arci] > head)
                right puth back (askCi]);
```

```
Std: Sort (lebt begin (), lebt. End ());
stal: sort (right begin (), right end());
int run=2;
while (run --) }
      if (derection = = 'lebt') {
         for Cint i = left . singe () -1; i>=0;
          i --) S
              westrack = left [i];
             sell-sequence. push-back (cur-track)
             distance = abs (cur_mack - head);
             Seek_count += distance;
             nead = cur track;
          direction = "right";
     else if (direction, z z "right") {
         for (inti=0; i < right. Size (); i++) {
            un track = right [i];
            Seek-Sequence. push-back (in track);
            distance = abs (cur rach-head);
            Seek_count + = distance;
            head = cue - track;
          direction =" left";
```

```
cout < ( Fotal number of seek operations = "
        << seek - court << endl;
    Lout CC" Seek lequence is " LC and!
    for Cint i = 0; i < seek-sequence. Size ();
        1++)8
        wut CK Seek-Seauonee [i] Kondi;
Int main ()
  int arr[size] = { 176, 79, 34, 60, 92, 11, 41,
   1144;
   int head = 30;
   Etring direction = "left";
   SCAN ( arr, head, direction);
   return 9
```

```
Q4
                          C-SCAN disk 8 cheduling
     C++ code for
      algorithm
     # include < bets | stdc++.h>
A.
     Using namespace etd;
     int size = 8;
    int disk size = 200;
    void CSCAN (int arr[], int head)
        ent seek_count =0;
        int distance, un track;
        vector<int> left, right,
        vector < int > seek_ sequence;
        left push back (0);
        right push_back(disk_Size -1);
        for Cint i = 0; in < size; i++) {
            of Carrlis ( head)
              left push back Carr[i]);
            y (arr[i] > head)
               right push back (arr [i]);
         Std: Sort (left begin (), left end ());
         Std: Sort (right begin(), right. end());
```

```
for Cint i=0; i < right line (); i++) of
   weitrack = right [i];
   Seek-Seavence push-back Cour-track);
   distance = abs (cur_track - head);
   Seek_count + = distance > > / de la la la
   head = curtrack;
                          C) DU MALIEUS DELLA
head = 0;
for (int i=0; i<left size(); i++) {
    us_track = left [i];
    Seek-Sequence push-back (cur-track);
    distance = abs ( cur-track - head );
    seek count + = distance;
    head = cur-track;
cout << 'Total number of seek operations="
     LL seek count LL endl;
cout ((" seek séquence is " ( L end!)
for Cint i = 0; 1/ < seek-sequence. size ();
i++) $
      cout << sacressor seek sequence [i]
          LC endl;
```

Scanned by CamScanner

```
int main ()
       int arr[size] = & 176, 79, 34, 60, 92, 11, 41,
                       1143;
       cout << "Initial position of head: "<< head
       LL endl;
       CSCAN (arr, head);
       return Oi
\Phi_5
      C++ code for LOOK disk scheduling
      problem.
A.
     dood
     #include < bits / stdc++ h>
     using namespace Stol;
     int line = 8;
     int disk_size $2,00;
     void LOOK/C int arrlJ, int head, string
      direction)
        int seek count = 0;
        int distance, un-track;
        vector < int > left, right;
        vector < int > seek-sequence;
```

```
ger (int i=0; i < size; i++) }
     if Carr Ci] < head)
        left. presh-back Care [i]);
    is (arrCi) > head)
        right push-back (arrei]);
Std: : Sort ( lebt. begin (), left. end());
Std:: sort ( right. begin (), right. end ());
int runz 2;
while (run --) }
      if (direction = 2" left") {
        for Cit i = left Size()-1; i> =0; i--){
            ur rack z lest [i];
            Seek- Sequence. push-back (cur_track);
            distance = abs (cur track - head),
            seek - went + = distance;
            head = aur_track;
         direction z" color right";
     The is (direction = = right)
        for Cint i zo; i < right size(); i++), I
            cur_track = right [i];
            seek seavence push back (au track);
            distance = abs (curstrack - head);
            seek_count += distance;
            head = au -track;
    ¿ direction =" lebt";
                                         PTO >
```

```
Lout << "Total number of seek operations = "
        LL Seek went ( sidl;
    cont << seeka sequence is "<< end;
     for Cint i 20; i < seek_ sequence. size (); i++)
         E E ANTIH BUILD
       went 26 seek_ Sequence [i] << and;
int main ()
 int exc[size]=$ 176,79,34,60,92,11,41,1143;
 Int head = 50;
  String direction = "right";
 cout << "Initial position of head" << head
      K endl;
  LOOK (arr, head, direction);
  return 0;
```

```
QL
      C++ program for C-cook dish scheduling
      algorithm
     #include < bits / etdc++.h>
A.
     using namespace stol;
     int size = 8;
int disk_size = 200;
     void CLOOK (int assII, int head)
         int seek count =0;
         int distance, our track;
         vector ¿int> left, right;
         elector (int) seek_sequence;
         fer ( int i = 0 +, i < singe; i ++) &
             if (arr[1] < head)
                 lebt. push -back (arr [i]);
             if (arrCi] > hoad)
                 right. bush_back (anti]);
          Std: sort (left. begin (), left. end());
          std: sort (right begin (), right end ()));
          for Cuit i=0; i < right size(); i++) of
              custrack = right Ci ];
              seek_ Sequence push back (cur track);
              distance = ass ( cur_track - head);
              Seek count + z distance;
              head = eur track;
                                             PTO
```

```
seek_count_= abs(head-left[6]);
   head = left [o];
   for (int i =0; i < left. Size (); i +1) }
       autrack = left [i];
      Seek Sequence push back (cue track);
      distance abs (curtrack - head);
      Seek_count_ = distance;
      head = un track;
    cout CC" Total number of seek operations z."
         Ch seek wint LC endl;
    cont L( " Sequence is " LC endl;
    for Cirt i 20; i ( Seek-Souence. Size (); i +4)
       west Whek seavence SiJ Kende;
int main ()
  int anseize] = {176, 79, 34, 60, 92, 11,
   41, 114 4;
   int head = 50
   coutce Initial position of head: ' < head (Lendly
   CLOOK (arr, head)
   return 0;
```