PSEUDOCODE:-

Return Type:- Double Dimensional Array / List Of Lists

```
vector<vector<int>> threesum( int arr[] , int n)                    //returning a 2D Array
{
   if( size of array < 3)
        return (an empty 2D array);

 (2D array to store answer)
 vector<vector<int>> ans;


 sort(arr, arr+n);
 // sorting the given array
for(int i=0; i<n; i++)
{
   int j = i+1;
   int k = n-1;
   while(j<k && j<n)
  {
    if(arr[j] + arr[k] == (-arr[i]) )                              // a+b = (-c)  ⇒ a+c+b = 0
   {
     ans.push_back({arr[i], arr[j], arr[k]});


     while(k!=0 && arr[k-1] == arr[k])
                k-=1;
     while(j!=n-1 && arr[j] == arr[j+1])
                j+=1;

     j++; k–;
     }
  else if(arr[j] + arr[k] > (-arr[i]) )
   {
         while(k!=0 && arr[k-1] == arr[k])
                k-=1;
         k-=1;
   }
else if(arr[j] + arr[k] < (-arr[i]) )
```

```
{
      while(j!=n-1 && arr[j] == arr[j+1])
                j+=1;
        j+=1;
}
}

   while(i!=n-1 && arr[i] == arr[i+1])
                i+=1;
        i+=1;
}

//now we have obtained the triplets we can simply sort the individual triplets

for(int i=0; i<n; i++)
{
  sort(ans[i].begin(), ans[i].end());
}
// a<=b && b<=c

return ans;
}
```

TIME COMPLEXITY :- O(n^2)
SPACE COMPLEXITY :- O(n)