15B17CI371 – Data Structures Lab
ODD 2024
Week 4-LAB A
Practice Lab

1.

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cout<<"Input the number of elements: ";
   cin>>n;
   int *arr=new int[n];
   cout<<"Input the elements: ";
   for(int i=0;i<n;i++)
       cin>>arr[i];
       int key;
       cout<<"Input the number to be searched : ";
       cin>>key;
   for(int i=0;i<n;i++)
       if(arr[i]==key)
          cout<<"Element found at index "<<i<<endl;
}
```

```
Input the number of elements: 7
Input the elements: 4 7 7 3 8 2 7
Input the number to be searched : 7
Element found at index 1
Element found at index 2
Element found at index 6
```

2.

```cpp
#include <iostream>
using namespace std;
int main()
{
   int n;
   cout<<"Input the number of elements: ";
   cin>>n;
   int *arr=new int[n];
   cout<<"Input the elements: ";
   for(int i=0;i<n;i++)
       cin>>arr[i];
   int prod;
   cout<<"Input the product : ";
   cin>>prod;
   bool pairFound=false;
   for(int i=0;i<n;i++)
       for(int j=i+1;j<n;j++)
           if(arr[i]*arr[j]==prod)
           {
               cout<<"Pair Found:("<<arr[i]<<", "<<arr[j]<<")"<<endl;
               pairFound=true;
               break;
           }
   if(!pairFound)
       cout<<"No pair found"<<endl;
}
```

```
Input the number of elements: 7
Input the elements: 2 3 4 7 8 9 11
Input the product : 77
Pair Found:(7, 11)
```

3.

```cpp
#include <iostream>
using namespace std;
void bubblesort(int arr[], int n) {
for (int i = 0; i < n - 1; ++i) {
for (int j = 0; j < n - i - 1; ++j) {
if (arr[j] > arr[j + 1]) {
int temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;
}
}
}
}
void wavesort(int arr[], int n) {
bubblesort(arr, n);
for (int i = 0; i + 1 < n; i += 2) {
int temp = arr[i];
arr[i] = arr[i + 1];
arr[i + 1] = temp;
}
}
int main() {
const int size = 7;
int arr[size] = {10, 90, 49, 2, 1, 5, 23};
wavesort(arr, size);
cout << "Wave-like array: ";
for (int i = 0; i < size; ++i) {
cout << arr[i] << " ";
}
cout << endl;
return 0;
}
```

```
Wave-like array: 2 1 10 5 49 23 90
```

4.

```
#include <iostream>
#include <algorithm>
using namespace std;
int binarySearch(const int arr[], int size, int key) {
int left = 0;
int right = size - 1;
while (left <= right) {
int mid = left + (right - left) / 2;
if (arr[mid] == key) {
return mid; // Key found
}
if (arr[mid] < key) {
left = mid + 1;
} else {
right = mid - 1;
}
}
return -1;
}
void findAllOccurrences(int arr[], int size, int key) { sort(arr, arr + size);
int index = binarySearch(arr, size, key);
if (index == -1) {
cout << "Element not found in the array" << endl; return;
}
int leftIndex = index;
while (leftIndex >= 0 && arr[leftIndex] == key) { cout << "Element found at index " <<
leftIndex << endl; leftIndex--;
}
int rightIndex = index + 1;
while (rightIndex < size && arr[rightIndex] == key) { cout << "Element found at index
" << rightIndex << endl; rightIndex++;
}
}
int main() {
int arr[] = {16, 31, 15, 27, 9, 15, 39, 15, 17, 12}; int size = sizeof(arr) /
sizeof(arr[0]);
int key = 31;
findAllOccurrences(arr, size, key);
return 0;
}
```

```
archittiwari@Archits-MacBook-Air DSA % cd "/Users/archittiwari/Desktop/DSA/" &&
Element found at index 8
```

```cpp
#include <iostream>
using namespace std;
void insertionSort(int arr[], int size) {
for (int i = 1; i < size; ++i) {
int key = arr[i];
int j = i - 1;
while (j >= 0 && arr[j] > key) {
arr[j + 1] = arr[j];
--j;
}
arr[j + 1] = key;
}
}
bool binarySearch(const int arr[], int size, int key) {
int left = 0;
int right = size - 1;
while (left <= right) {
int mid = left + (right - left) / 2;
if (arr[mid] == key) {
return true;
}
if (arr[mid] < key) {
left = mid + 1;
} else {
right = mid - 1;
}
}
return false;
}
void findPairWithProduct(int arr[], int size, int n) { insertionSort(arr, size);
for (int i = 0; i < size; ++i) {
if (arr[i] == 0) {
continue;
}
if (n % arr[i] == 0) {
int complement = n / arr[i];
if (binarySearch(arr, size, complement)) {
cout << "Pair Found: (" << arr[i] << ", " << complement << ")" << endl; return;
```

```
}
}
}
cout << "No pair found" << endl;
}
int main() {
int arr[] = {5, 20, 3, 2, 50, 80};
int size = sizeof(arr) / sizeof(arr[0]);
int n = 100;
findPairWithProduct(arr, size, n);
return 0;
}
```

```
architttiwari@Architts-MacBook-Air DSA % cd "
Pair Found: (2, 50)
architttiwari@Architts-MacBook-Air DSA %
```

```
#include <iostream>
using namespace std;
void insertionSort(int arr[], int size) {
for (int i = 1; i < size; ++i) {
int key = arr[i];
int j = i - 1;
while (j >= 0 && arr[j] > key) {
arr[j + 1] = arr[j];
--j;
}
arr[j + 1] = key;
}
}
bool interpolationSearch(const int arr[], int size, int key) { int left = 0;
int right = size - 1;
while (left <= right && key >= arr[left] && key <= arr[right]) { if (left == right) {
if (arr[left] == key) return true;
return false;
}
int pos = left + ((key - arr[left]) * (right - left)) / (arr[right] - arr[left]);
if (arr[pos] == key) {
return true;
}
```

```
if (arr[pos] < key) {
left = pos + 1;
} else {
right = pos - 1;
}
}
return false;
}
void findPairWithProduct(int arr[], int size, int n) {
insertionSort(arr, size);
for (int i = 0; i < size; ++i) {
if (arr[i] == 0) {
continue;
}
if (n % arr[i] == 0) {
int complement = n / arr[i];
if (interpolationSearch(arr, size, complement)) {
cout << "Pair Found: (" << arr[i] << ", " << complement << ")" << endl; return;
}
}
}
cout << "No pair found" << endl;
}
int main() {
int arr[] = {5, 20, 3, 2, 50, 80};
int size = sizeof(arr) / sizeof(arr[0]);
int n = 4000;
findPairWithProduct(arr, size, n);
return 0;
}
```

```
architttiwari@Archits-MacBook-Air DSA % cd "/U
Pair Found: (50, 80)
architttiwari@Archits-MacBook-Air DSA %
```

Virtual Labs