Atma Ram Sanatan Dharma College
University of Delhi

Discrete Structures Practical

Paper Code: 32341202

Submitted By
Sadhna
Roll No: 88059
Sem: II
B.Sc. Computer Science(Hons)

Submitted To
Ms. Shalini Gupta
Department of Computer Science

Q5. WAP generate Fibonacci series using recursion.

PROGRAM:

```
#include
using namespace std;
int fib(int x) {
  if((x==1)||(x==0)) {
    return(x);
  }else {
    return(fib(x-1)+fib(x-2));
  }
}
int main() {
  int x , i=0;
  cout << "Enter the number of terms of series : ";
  cin >> x;
  cout << "\nFibonnaci Series : ";
  while(i < x) {
    cout << " " << fib(i);
    i++;
  }
  return 0;
}
```

Output:

```
Enter the number of terms of series : 15
Fibonnaci Series : 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Q6. WAP to implement Tower of Hanoi using recursion.

PROGRAM:

```
#include<iostream>
using namespace std;
//tower of HANOI function implementation
void TOH(int n,char Sour, char Aux,char Des)
{
        if(n==1)
        {
                cout<<"Move Disk "<<n<<" from "<<Sour<<" to "<<Des<<endl;
                return;
        }

        TOH(n-1,Sour,Des,Aux);
        cout<<"Move Disk "<<n<<" from "<<Sour<<" to "<<Des<<endl;
        TOH(n-1,Aux,Sour,Des);
}
//main program
int main()
{
        int n;
        cout<<"Enter no. of disks:";
        cin>>n;
        //calling the TOH
        TOH(n,'A','B','C');
        return 0;
}
```

Q7. WAP to implement binary search using recursion.

PROGRAM

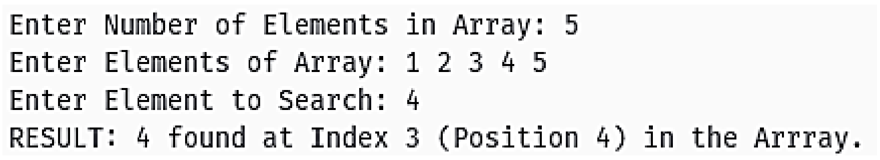
```
#include <iostream>

using namespace std;

int binarySearch(int *arr, int el, int first, int last)
{
    if (first <= last)
    {
        int mid = (first + last) / 2;
        if (el == arr[mid])
            return mid;
        else if (el < arr[mid])
            binarySearch(arr, el, first, mid - 1);
        else if (el > arr[mid])
            binarySearch(arr, el, mid + 1, last);
    }
    else
        return -1;
}

int main()
{
    int n;
    cout << "Enter Number of Elements in Array: ";
    cin >> n;
    int arr[n];
    cout << "Enter Elements of Array: ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];
    int el;
    cout << "Enter Element to Search: ";
    cin >> el;
    cout << "RESULT: ";
    int index = binarySearch(arr, el, 0, n - 1);
    if (index == -1)
        cout << el << " was not found in the Array.";
    else
        cout << el << " found at Index " << index
             << " (Position " << (index + 1)
             << ") in the Arrray.";
    return 0;
}
```

Output



```
Enter Number of Elements in Array: 5
Enter Elements of Array: 1 2 3 4 5
Enter Element to Search: 4
RESULT: 4 found at Index 3 (Position 4) in the Arrray.
```

Q8. Write a Program to implement Bubble Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.

PROGRAM:

```
#include <cstdlib>
#include <fstream>
#include <iostream> using
namespace std;
int bubbleSort(int *, int);int main()
{
        int size, comparisons;
        cout << "Enter array size: ";cin >> size;
        int array[size];
        // Worst Case
        cout << "Worst Case:\n------------------------------\n";
        for (int i = 0; i < size; i++)array[i] = size -
                i;
        comparisons = bubbleSort(array, size);cout <<
        "Total Comparisons Made: "
                  << comparisons << endl
                  << endl;
        // Best Case
        cout << "Best Case:\n ------------------------------ \n";
        for (int i = 0; i < size; i++)array[i] = i + 1;
        comparisons = bubbleSort(array, size);cout <<
        "Total Comparisons Made: "
                  << comparisons << endl
                  << endl;
        // Average Case
        cout << "Average Case:\n------------------------------\n";
        ifstream fin("./random.txt"); for (int i =
        0; i < size; i++)
fin >> array[i];fin.close();
            comparisons = bubbleSort(array, size);cout
            << "Total Comparisons Made: "
                   << comparisons << endl
                   << endl;

            return 0;
      }
      int bubbleSort(int *array, int size)
      {
            int temp, iterCompCount, count = 0;
            cout << "Array: ";
            for (int j = 0; j < size; j++)cout <<
                  array[j] << " ";
            cout << endl;
            for (int i = 0; i < size - 1; i++)
            {
                  iterCompCount = 0;
                  for (int j = 0; j < size - i - 1; j++)
                  {
                        if (array[j + 1] < array[j])
                        {
                              temp = array[j]; array[j]
                              = array[j + 1];array[j + 1]
                              = temp;
                        }
                        count++;
                        iterCompCou
                        nt++;
                  }
                  cout << "Comparisons Made in Pass " << i + 1
                         << ": " << iterCompCount << endl;
                  cout << "After Pass " << i + 1 << ": ";for (int k
                  = 0; k < size; k++)
                        cout << array[k] << " ";
                  cout << endl;
            }
            return count;
      }
```

Q9. Write a Program to implement Insertion Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.

PROGRAM:

```
#include <cstdlib> #include <fstream> #include <iostream> using namespace std;
int insertionSort(int *, int); int main()
{
int size, comparisons;
cout << "Enter array size: "; cin >> size;
int array[size];
// Worst Case
cout << "Worst Case:\n \n";
for (int i = 0; i < size; i++) array[i] = size - i;
comparisons = insertionSort(array, size); cout << "Total Comparisons Made: "
<< comparisons << endl
<< endl;
// Best Case
cout << "Best Case:\n    \n";
for (int i = 0; i < size; i++) array[i] = i + 1;
comparisons = insertionSort(array, size); cout << "Total Comparisons Made: "
<< comparisons << endl
<< endl;
// Average Case
cout << "Average Case:\n          \n";
ifstream fin("./random.txt"); for (int i = 0; i < size; i++)

fin >> array[i]; fin.close();
comparisons = insertionSort(array, size); cout << "Total Comparisons Made: "
<< comparisons << endl
<< endl; return 0;
}
int insertionSort(int *array, int size)
{
int i, j, k, key, iterCompCount, count = 0; cout << "Array: ";
for (k = 0; k < size; k++) cout << array[k] << " ";
cout << endl;
for (i = 1; i < size; i++)
{
key = array[i]; iterCompCount = 0;
for (j = i - 1; j >= 0; j--)
{
count++; iterCompCount++; if (array[j] > key)
{
array[j + 1] = array[j];
}
else
{
break;
}
}
array[j + 1] = key;
cout << "Comparisons Made in Pass " << i << ": "
<< iterCompCount << endl;
cout << "After Pass " << i << ": "; for (k = 0; k < size; k++)
cout << array[k] << " "; cout << endl;
}
return count;
}
```

==