CMPT 130 Week 13 Lab Work

1. Consider the following implementation of the sequential search. What will be the output of the program if the value of x is 0?

```
int seqSearch(const int A[], const int startIndex, const int lastIndex, const int searchValue)
{
    if (startIndex > lastIndex)
           return -1;
    else if (A[startIndex] == searchValue)
           return startIndex;
    else
    {
           for (int i = startIndex + 1; i <= lastIndex; i++)
                  cout << A[i] << "   ";
           cout << endl;
           return seqSearch(A, startIndex+1, lastIndex, searchValue);
    }
}
int main()
{
    const int size = 8;
    int A[size] = {2, 8, 3, 7, 9, 0, 1, 6};
    int x;
    cout << "Enter an integer number to search in the array ";
    cin >> x;
    int index = seqSearch(A, 0, size-1, x);
    if (index == -1)
           cout << x << " is not found in the array." << endl;
    else
           cout << x << " is found in the array at index " << index << endl;
    system("pause");
    return 0;
}
```

2. Consider the following implementation of the sequential search algorithm. What will be the output of the program if the value of x is 1?

```
int seqSearch(const int A[], const int startIndex, const int lastIndex, const int searchValue)
{
    if (startIndex > lastIndex)
           return -1;
    else if (A[startIndex] == searchValue)
           return startIndex;
    else
    {
           int index = seqSearch(A, startIndex+1, lastIndex, searchValue);
           for (int i = startIndex + 1; i <= lastIndex; i++)
                  cout << A[i] << "   ";
           cout << endl;
           return index;
    }
}

int main()
{
```

```
    const int size = 8;
    int A[size] = {2, 8, 3, 7, 9, 0, 1, 6};
    int x;
    cout << "Enter an integer number to search in the array ";
    cin >> x;
    int index = seqSearch(A, 0, size-1, x);
    if (index == -1)
           cout << x << " is not found in the array." << endl;
    else
           cout << x << " is found in the array at index " << index << endl;
    system("pause");
    return 0;
}
```

3. What is the output of the following program if the value of x is 20?

```
int binarySearch(const int A[], const int startIndex, const int lastIndex, const int searchValue)
{
    if (startIndex > lastIndex)
           return -1;
    else
    {
           int m = (startIndex + lastIndex) / 2;
           if (A[m] == searchValue)
                  return m;
           else if (A[m] > searchValue)
           {
                  for (int i = startIndex; i < m; i++)
                         cout << A[i] << "  ";
                  cout << endl;
                  return binarySearch(A, startIndex, m-1, searchValue);
           }
           else
           {
                  for (int i = m+1; i <= lastIndex; i++)
                         cout << A[i] << "  ";
                  cout << endl;
                  return binarySearch(A, m+1, lastIndex, searchValue);
           }
    }
}
int main()
{
    const int size = 8;
    int A[size] = {3, 7, 12, 17, 21, 25, 30, 35};
    int x;
    cout << "Enter an integer number to search in the array ";
    cin >> x;
    int index = binarySearch(A, 0, size-1, x);
    if (index == -1)
           cout << x << " is not found in the array." << endl;
    else
           cout << x << " is found in the array at index " << index << endl;
    system("pause");
    return 0;
}
```

4. What is the output of the following program if the value of x is 20?

```
int binarySearch(const int A[], const int startIndex, const int lastIndex, const int searchValue)
{
```

```
    if (startIndex > lastIndex)
            return -1;
    else
    {
            int m = (startIndex + lastIndex) / 2;
            if (A[m] == searchValue)
                    return m;
            else if (A[m] > searchValue)
            {
                    int index = binarySearch(A, startIndex, m-1, searchValue);
                    for (int i = startIndex; i < m; i++)
                            cout << A[i] << "  ";
                    cout << endl;
                    return index;
            }
            else
            {
                    int index = binarySearch(A, m+1, lastIndex, searchValue);
                    for (int i = m+1; i <= lastIndex; i++)
                            cout << A[i] << "  ";
                    cout << endl;
                    return index;
            }
    }
}
int main()
{
    const int size = 8;
    int A[size] = {3, 7, 12, 17, 21, 25, 30, 35};
    int x;
    cout << "Enter an integer number to search in the array ";
    cin >> x;
    int index = binarySearch(A, 0, size-1, x);
    if (index == -1)
            cout << x << " is not found in the array." << endl;
    else
            cout << x << " is found in the array at index " << index << endl;
    system("pause");
    return 0;
```

5. Write a recursive void function that has one parameter which is a positive integer **n** and that prints **n** asterisks '*****' to the screen all on one line.

6. Write a recursive function that takes a positive integer argument **n** and prints, in the order give, the integers **n n-1 n-2 . . . 2 1**

7. Write a recursive function that takes a positive integer argument **n** and prints, in the order given, the integers **1 2 3 . . . n-1 n**

8. Write a recursive function named **recursiveSum** that takes a positive integer argument **n** and returns the sum, in the order given, of the integers **n + (n-1) + (n-2) + . . . 3 + 2 + 1**.

9. Write a recursive function named **recursiveSumDifficult** that takes a positive integer argument **n** and returns the sum, in the order given, of the integers **1 + 2 + 3 + ... + (n-2) + (n-1) + n**.

10. Write a C++ recursive function named **reversePrint** that takes a non-negative integer argument and prints the number in reverse. For example: if the argument integer is 1234, then the function must print **4321**

11. Observe that given any integer **x** and a positive integer **y**, the expression $\mathbf{x^y}$ can be written as $\mathbf{x*x^{y-1}}$. Write a C++ recursive function named power that takes an integer **x** and a potive integer **y** and returns **x raised to the power of y**.

12. Write a recursive function named **squares** that takes a positive integer argument **n** and returns the sum, in the order given, of the squares $\mathbf{n^2 + (n-1)^2 + (n-2)^2 + \ldots + 1^2}$. For example, **squares(3)** returns **14** because $\mathbf{3^2 + 2^2 + 1^2}$ is **14**.

13. The mathematical Fibonacci sequence is given by $\boldsymbol{fib(n)} = \begin{cases} \mathbf{1} & \boldsymbol{if\ n = 0} \\ \mathbf{1} & \boldsymbol{if\ n = 1} \\ \boldsymbol{fib(n-1) + fib(n-2)} & \boldsymbol{if\ n > 1} \end{cases}$

Write a recursive function that takes a non-negative integer argument **n** and returns ***fib(n)***.

The critical operation of the algorithm is the addition operation. The complexity of the algorithm for a given value **n** is therefore

f(n) = number of times the addition operation is performed to calculate ***fib(n)***

What is the **Big-O** of ***f(n)***?

14. Although at times hard, every recursive algorithm can be replaced with a non-recursive algorithm. Modify your recursive Fibonacci function in Question #18 above so that it uses a non-recursive algorithm.

15. Have fun! Which of the programs between Question #18 and Question #19 is faster? Why?

16. The mathematical Ackerman function is given by

$$\boldsymbol{A(m,n)} = \begin{cases} \boldsymbol{n+1} & \boldsymbol{if\ m = 0} \\ \boldsymbol{A(m-1,1)} & \boldsymbol{if\ m > 0}\ and\ n = 0 \\ \boldsymbol{A(m-1, A(m, n-1))} & \boldsymbol{if\ m > 0}\ and\ n > 0 \end{cases}$$

Write a recursive function that takes two non-negative integer arguments **m** and **n** and returns A(m,n). What is the Big-O of A(m,n)?

17. Write a recursive C++ function named **cStringSequentialSearch** that takes a c-string, a starting index, and a character as arguments and returns true if the character is found in the c-string otherwise returns false. Test your function by writing a main program where you create a c-string and a character and calling the function as **bool ans = cStringSequentialSearch (C, 0, ch);** where C is your cstring, ch is a character variable initialized with some character value to search in your cstring and of course start the search at index 0.

18. Write a recursive C++ function named **cStringBinarySearch** that takes a cstring, a starting index, last index, and a character as arguments and returns true if the character is found in the cstring otherwise returns

false. Assume the cstring is sorted. Test your function by writing a main program where you create a sorted cstring and a character and then calling the function as

bool ans = cStringBinarySearch(C, 0, len-1, ch); //len is the length of the cstring C

where C is your sorted cstring, ch is a character variable initialized with some character value to search in your cstring and of course start the search at index 0.

19. Write a recursive function that takes an array of double data type, a start index, and a last index and returns the maximum element of the array.

20. Write a recursive function that takes an array of integers, a start index and a last index and prints the elements of the array in reverse order.

21. Write a recursive function that takes a cstring, a start index and last index and reverses the cstring.

22. Write a recursive function that takes an array of float data types, a start index, and a last index and checks if the elements of the array are sorted in increasing order. If yes, return true; otherwise return false.

23. A recursive increasing order bubble sort can be thought of as follows: If the array to be sorted has one or less elements then do nothing. Otherwise first bubble up the largest element to the last position and then bubble sort the array excluding the last element. Write a recursive C++ function named **recursiveBubbleSort** that takes an array and its size and sorts the array using bubble sort recursively.

24. A recursive increasing order insertion sort can be thought of as follows: If the array to be sorted has one or less elements then do nothing. Otherwise first insertion sort the array excluding the last element then insert the last element in its right location. Write a recursive C++ function named **recursiveInsertionSort** that takes an array and its size and sorts the array using insertion sort recursively.

25. A recursive increasing order selection sort can be thought of as follows: If the array to be sorted has one or less elements then do nothing. Otherwise first select the largest element and put it at the last position and then selection sort the array excluding the last element. Write a recursive C++ function named **recursiveSelectionSort** that takes an array and its size and sorts the array using selection sort recursively.