# Recursion and Dynamic Memory allocation

## Introduction:

In this lab we will practice input and output file stream, recursion, and dynamic memory allocation. Moreover, for now on we will not the computer display as output i.e. `cout` is prohibited. You can use file stream to log any output or use a file to read any input instead of keyboard input i.e. `cin`. You can use either command line compiler installed in your system from visual studio or an online compiler. If you use an online compile make sure you save your source code in a local folder. You should study the code and answer the question given in the exercise.

**Instruction:** Use proper indentation file writing your code. Your marks during the exams will be deducted if proper indentation is not used. Also write comments in your code as necessary, especially at the start of the program file. Save your source files in a separate folder and name files and folder accordingly. Do not copy the code from your piers you can always ask your instructors to help you finish the lab exercise.

## File stream

Run the code and answer the question give in the exercise. Following C++ program opens a file in reading and writing mode. After writing information entered by the user to a file named afile.dat, the program reads information from the file and outputs it onto the screen

### Code-1

```cpp
#include <fstream>
#include <iostream>
using namespace std;

int main () {
   char data[100];

   // open a file in write mode.
   ofstream outfile;
   outfile.open("afile.dat");

   cout << "Writing to the file" << endl;
   cout << "Enter your name: ";
   cin.getline(data, 100);

   // write inputted data into the file.
   outfile << data << endl;

   cout << "Enter your age: ";
   cin >> data;
   cin.ignore();

   // again write inputted data into the file.
   outfile << data << endl;

   // close the opened file.
   outfile.close();

   // open a file in read mode.
```

```
  ifstream infile;
  infile.open("afile.dat");

  cout << "Reading from the file" << endl;
  infile >> data;

  // write the data at the screen.
  cout << data << endl;

  // again read the data from the file and display it.
  infile >> data;
  cout << data << endl;

  // close the opened file.
  infile.close();

  return 0;
}
```

**Exercise-1**

1. What happens if you change the file extension?
2. What happens if you do not use `cin.ignore();`?
3. What is the benefit of using `cin.getline(data, 100);` instead of `cin>>data` ?
4. Modify the program to enter a list of 10 students, their age and gender.

## Recursion

Recursive functions are functions that call themselves (directly, or indirectly through another function). In the lectures we studied a simple example to learn recursion. We saw that recursion makes the code more simpler and elegant. Two things that pretty much all correct recursive functions share are the following:

1. A recursive function needs one or more base case.
2. Recursive calls must have smaller inputs than the main input.

Here we study some more complex example to understand recursion.

**Problem:** Given a pre-sorted array of integers in increasing order, find the location of a target element, or return -1 if it is not in the array.

**Solution: Binary Search algorithm:** Check the middle of the remaining array. If the element is there, we are done. If the desired element is smaller, continue searching to the left of the middle element; otherwise, continue searching to the right

**Code-2a**

```
//  Created by Frank M. Carrano and Tim Henry.

/** Searches the array anArray[first] through anArray[last]
    for a given value by using a binary search.
 @pre  0 <= first, last <= SIZE - 1, where SIZE is the
    maximum size of the array, and anArray[first] <=
    anArray[first + 1] <= ... <= anArray[last].
 @post  anArray is unchanged and either anArray[index] contains
    the given value or index == -1.
 @param anArray  The array to search.
 @param first  The low index to start searching from.
 @param last  The high index to stop searching at.
 @param target  The search key.
 @return  Either index, such that anArray[index] == target, or -1.
```

```
*/
int binarySearch(const int anArray[], int first, int last, int target)
{
   int index;
   if (first > last)
      index = -1; // target not in original array
   else
   {
      // If target is in anArray,
      // anArray[first] <= target <= anArray[last]
      int mid = first + (last - first) / 2;
      if (target == anArray[mid])
         index = mid; // target found at anArray[mid]
      else if (target < anArray[mid])
         // Point X
         index = binarySearch(anArray, first, mid - 1, target);
      else
         // Point Y
         index = binarySearch(anArray, mid + 1, last, target);
   } // end if

   return index;
} // end binarySearch
```

### 0.0.1 Exercise-2a

1. Find an iterative solution to the above binary search
2. 
3. Find the two properties of the recursive solution in above code
4. Draw a data-flow or box traces of the above recursive call for a given array $A = 1, 5, 9, 12, 15, 21, 29, 31$ and targets $target = 9$ and $target = 6$

**Code-2b**

The following function computes the sum of the first $n \geq 1$ integers.

```
//  Created by Frank M. Carrano and Tim Henry.

/** Computes the sum of the integers from 1 through n.
 @pre  n > 0.
 @post  None.
 @param n  A positive integer.
 @return  The sum 1 + 2 + . . . + n. */
int sumUpTo(int n)
{
   int sum = 0;
   if (n == 1)
      sum = 1;
   else // n > 1
      sum = n + sumUpTo(n - 1);
   return sum;
} // end sumUpTo
```

**Exercise-2b**

1. Show how this function satisfies the properties of a recursive function
2. Write the recursive solution to above problem
3. Draw a block trace or data flow of the above code for $n = 4$

### 0.0.2 Code-2c

The following code prints a string backward

```
//  Created by Frank M. Carrano and Tim Henry.
//  Copyright (c) 2013 __Pearson Education__. All rights reserved.

/** Iterative version. */
void writeBackward(string s)
{
   int length = (int)s.size();
   while (length > 0)
   {
      cout << s.substr(length - 1, 1);
      length--;
   }  // end while
}  // end writeBackward
```

#### Exercise-2c

1. Show how this function satisfies the properties of a recursive function.
2. Write a recursive solution to above problem and draw a box trace or data flow for input string *"rat"*.
3. Use the function to print strings of a fixed length.

#### Code-3

Run the following code and answer the following:

```
//dynamic allocation
#include<iostream>
using namespace std;
int main()
{
  int* ptr = new int;
  *ptr = 2;
  *ptr = *ptr * 4;
  cout << "Data at memory is " << *ptr << endl;
  delete ptr;
  ptr = NULL;
  return 0;
}
```

#### Exercise-3

1. Can you write the C-style memory allocation for above code?
2. Make an array of students in the class along with their name, age, and gender data. Let the user define the size of the array instead of a fixed size array. Read the name from the file created in exercise-1 into the array and print it on the display screen.
3. Suppose one student has left the class. Can you write a function to delete the name along with its age and gender from the array?

## References

1. Tutorialspoint website 2017, `https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm`.
2. Lecture Notes Data Structure `http://www-bcf.usc.edu/\%7Edkempe/teaching/DataStructures.pdf`
3. Linkedlist tutorial `http://www.zentut.com/c-tutorial/c-linked-list/`