



LABORATORY LOGBOOK 2

Hamza Bhatti (21223241)

Contents

Introduction	3
Bootstrap Website	3
Pages	3
Code	5
Link	7
Java Exercises	8
Abstract Data Type.....	8
Code	8
Application	9
Execution.....	10
Searching.....	11
Application	11
Execution.....	12
Bracketing	13
Application	13
Execution.....	14
Autotools.....	15
Configure.ac	15
Makefile.am	15
Main.c	16
Student.c	16
Student.h.....	17
Makefile.am 2	17
Compiling	17
Debugging	18
Ex1.....	18
Solution	18
Execution.....	18
Ex2.....	19
Solution	19
Execution.....	19
Ex3.....	20
Solution	20
Execution.....	20

Ex4.....	21
Solution	21
Ex5.....	22
Solution	22
Execution.....	23
Ex6.....	24
Solution	24
Execution.....	26

Introduction

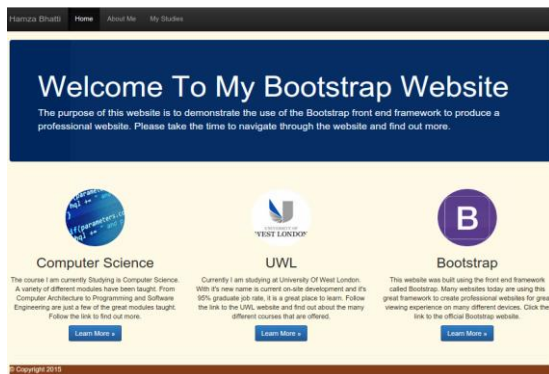
The purpose of this logbook is to produce a variety of programs in different areas of Computer Science. The activities include the development of a web page using the front end framework Bootstrap, some Java activities, the use of Autotools and a small number of debugging tasks.

Bootstrap Website

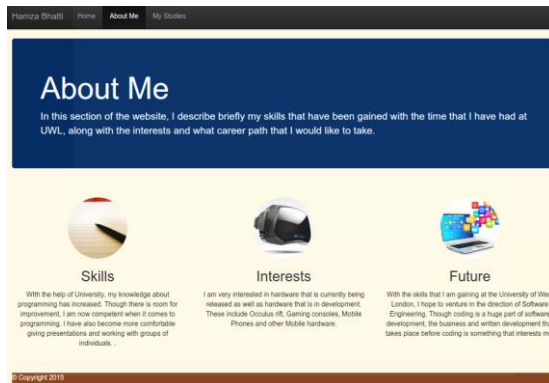
In the last Laboratory assignment, a basic HTML page was created, using standard headings and the use of simple CSS structure. With this new assignment, a somewhat more complex website was developed with the use of the front end framework called Bootstrap.

Pages

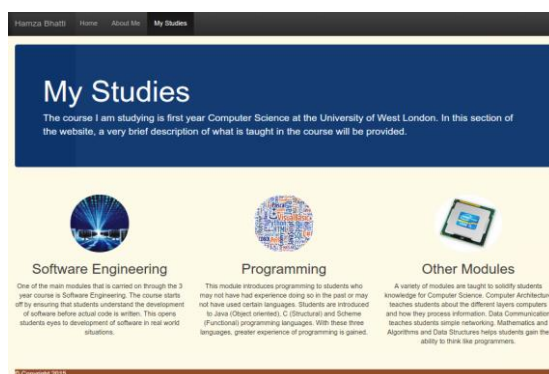
Home



About Me



My studies



As can be seen, all pages follow the same structure and colour scheme. The main differences are the heading in the Jumbotron header, images and the text for each heading.

The website developed used a fixed layout, and can really only viewed on a computer screen rather than a mobile or tablet. With this in mind, future developed Bootstrap website should evolve into a more fluid layout which can adjust the page when viewed on any screen for a better experience.

The bootstrap layout used was a mix of the carousel and Jumbotron templates. This mix was found with the use of rounded images above heading along with the main header of the webpage being the Jumbotron itself.

When coding the website, Classes where used for better structure. The classes helped identify which structures the following code belonged to. The CSS structure is shared throughout all the pages. But the CSS is contained within the HTML files themselves. This involved copy and paste of most of the code for the other pages, with changes to text and images.

Code

The same code is shared throughout all the pages. The following code given is from the index page only. Comments are included with the code to show what each class is associating with.

```
<!-- HTML page using Bootstrap front end framework -->

<!-- HOME PAGE -->

<!DOCTYPE html>
<html lang = "eng">

<!-- head element is a container for meta data/ data about the data -->
<head>
<meta charset = "UTF-8">
<title>Laboratory: HTML Using Bootstrap</title>

<!-- Importing the use of bootstrap and jquery -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap-theme.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>

<!-- Css section of HTML -->
<style type="text/css">
    <!-- For spacing between navbar and jumbotron -->
    html {
        width: 100%;
        height: 100%;
        background-color: #fff;
    }
    body {
        font-family: "Helvetica Neue",Helvetica,Arial,sans-serif;
        font-size: 14px;
        line-height: 1.42857143;
        color: #333;
        background-color: #FFF8DC;
    }
    .jumbotron{
        margin-top: 70px;
        color: white;
        background-color: #003366;
    }
    <!-- For image, heading and paragraph -->
    .center-block {
        display: block;
        margin: auto;
    }
    .col-xs-4{
        padding-top: 30px;
        background-color: #FFF8DC;
    }
    .col-xs-4 h2{
        text-align: center;
    }
    .col-xs-4 p{
        text-align: center;
    }
    <!-- For footer -->
    .col-xs-12{
        background-color: #834C24;
    }
    footer{
        float: left;
        width: 1139px;
        color: white;
        background-color: #834C24;
    }
</style>
</head>

<!-- body tag stating what is in the website -->
```

```

<body>

<!-- nav for defining navigation around the website -->
<nav id="myNavbar" class="navbar navbar-default navbar-inverse navbar-fixed-top" role="navigation">

    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="container">
        <!-- Creating the button arrangement -->
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target="#navbarCollapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand">Hamza Bhatti</a>
        </div>

        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="navbarCollapse">
            <ul class="nav navbar-nav">
                <!-- Li defines list items, active for current page, a for
hyper link -->
                <li class="active"><a href="#">Home</a></li>
                <li><a href="pages/aboutMe.html">About Me</a></li>
                <li><a href="pages/myStudies.html">My Studies</a></li>
            </ul>
        </div>
    </div>
</nav>

<div class="container">

    <!-- Main heading and info area -->
    <div class="jumbotron">
        <h1>Welcome To My Bootstrap Website</h1>
        <p>The purpose of this website is to demonstrate the use of the Bootstrap front end
framework to produce a professional website. Please take the time to navigate through the website and
find out more.</p>
    </div>

    <!-- Sections where other information will be displayed -->
    <div class="row">
        <div class="col-xs-4">
            
            <h2>Computer Science</h2>
            <p>The course I am currently Studying is Computer Science. A variety of
different modules have been taught. From Computer Architecture to Programming and Software Engineering
are just a few of the great modules taught. Follow the link to find out more.</p>
            <!-- Link to My Studies page -->
            <p><a href="pages/myStudies.html" class="btn btn-primary">Learn More
&raquo;</a></p>
        </div>

        <div class="col-xs-4">
            
            <h2>UWL</h2>
            <p>Currently I am studying at University Of West London. With it's new
name is current on-site development and it's 95% graduate job rate, it is a great place to learn. Follow
the link to the UWL website and find out about the many different courses that are offered.</p>
            <!-- Link to UWL website -->
            <p><a href="http://www.uwl.ac.uk/" target="_blank" class="btn btn-
primary">Learn More &raquo;</a></p>
        </div>

        <div class="col-xs-4">
            
            <h2>Bootstrap</h2>

```

`<p>`This website was built using the front end framework called Bootstrap. Many websites today are using this great framework to create professional websites for great viewing experience on many different devices. Click the link to the official Bootstrap website.`</p>`

```
<!-- Link to bootstrap website -->
<p><a href="http://getbootstrap.com/" target="_blank" class="btn btn-
primary">Learn More &raquo;</a></p>
</div>
<hr>

<!-- Acts as a footer -->
<div class="row">
  <div class="col-xs-12">
    <footer>
      <p>&copy; Copyright 2015</p>
    </footer>
  </div>
</div>
</div>
</body>
</html>
```

Link

The website can be found at this address:

<http://soc.uwl.ac.uk/~21223241/>

Java Exercises

Abstract Data Type

The first Java task was to create an abstract data type. The ADT created was a Person ADT. The constructor to make an object used parameters. The Accessors used here were `compareTo()` and `toString()` methods. The `compareTo()` method compares the age of this object to that object. An int value is returned. If the int is -1, this person is younger than that person. When the value is 1, this person is older than that person and if the value is 0, they are both the same age. The transformer implemented is a mutative transformer that changes the weight of a person.

Code

This code displays the creation of the ADT. Comments are included to identify what method is doing what.

```
public class PersonADT
{
    /*
     * Class declaration
     */

    // Data representation should be private
    private int age, yearOfBirth;
    private String fullName, nationality;
    private double height;
    private double weight;

    ////////// Constructor //////////

    public PersonADT (String fullName, int age, int yearOfBirth,
                      String nationality, double height,
                      double weight)
    {
        // Construct a person that has an age, year of birth,
        // full name, nationality, height(meters) and weight(stone)
        this.fullName = fullName;
        this.age = age;
        this.yearOfBirth = yearOfBirth;
        this.nationality = nationality;
        this.height = height;
        this.weight = weight;
    }

    ////////// Accessor //////////

    public int compareTo (PersonADT that)
    {
        // When comparing, return 1 if this person is older than,
        // that person, return -1 if this person is younger than,
        // that person, and 0 when they are equal

        //if (this.age > that.age)
        //    return 1;
        //if (this.age < that.age)
        //    return -1;
        //else
        //    return 0;
        return (this.age > that.age ? +1 :
                this.age < that.age ? -1 :
                this.age > that.age ? +1 : 0);
    }

    public String toString ()
    {
        // Give the details/ attributes of the person in
    }
}
```

```

        // String format
        return("Full Name: " + this.fullName + ", " + "Age: " + this.age
            + ", " + "Year of Birth: " + this.yearOfBirth + ", "
            + "Nationality: " + this.nationality + ", " +
            "Height(meters): " + this.height + ", " + "Weight(stone): " +
this.weight);
    }

    //////////// Transformer ////////////

    public void changeWeight (int n)
    {
        // change the weight of the person
        this.weight = n;
    }
}

```

Application

This code was used to apply the use of the ADT.

```

public class PersonTest
{
    /*
     * Person class application code
     */

    public static void main(String[] args)
    {
        //////////// Constructor ////////////

        // Constructing people using Person class
        PersonADT hamza = new PersonADT("Hamza Bhatti", 20, 1993, "British", 1.77, 11.00);
        PersonADT john = new PersonADT("John Doe", 28, 1988, "American", 1.80, 9.00);
        PersonADT homer = new PersonADT("Homer Simpson", 30, 1970, "American", 1.90, 17);
        PersonADT peter = new PersonADT("Peter Griffin", 30, 1970, "American", 1.90, 20);

        //////////// Accessor ////////////

        // Using the compareTo method homer and peter
        System.out.println("Comparing Homer's age with Peter's: ");
        if(homer.compareTo(peter) > 0 )
            System.out.println("Homer is older than Peter\n");
        else if (homer.compareTo(peter) < 0 )
            System.out.println("Homer is younger than Peter\n");
        else
            System.out.println("They are the same age\n");

        // Using toString method to print details out in string format
        System.out.println("Printing people:");
        System.out.println(hamza.toString() + "\n" );
        System.out.println(john.toString() + "\n" );
        System.out.println(homer.toString() + "\n" );
        System.out.println(peter.toString() + "\n" );

        //////////// Transformer ////////////

        // Using the changeWeight method
        // Original details
        System.out.println("Hamza's original weight:");
        System.out.println(hamza.toString() + "\n");
        // Changing weight
        System.out.println("Hamza's new weight");
        hamza.changeWeight(10);
        // Details with new weight
        System.out.println(hamza.toString() + "\n");
    }
}

```

Execution

The image displays the use of the ADT in testing circumstances.

```
Comparing Homer's age with Peter's:
They are the same age
```

```
Printing people:
```

```
Full Name: Hamza Bhatti, Age: 20, Year of Birth: 1993, Nationality: British, Height(meters): 1.77, Weight(stone): 11.0
```

```
Full Name: John Doe, Age: 28, Year of Birth: 1988, Nationality: American, Height(meters): 1.8, Weight(stone): 9.0
```

```
Full Name: Homer Simpson, Age: 30, Year of Birth: 1970, Nationality: American, Height(meters): 1.9, Weight(stone): 17.0
```

```
Full Name: Peter Griffin, Age: 30, Year of Birth: 1970, Nationality: American, Height(meters): 1.9, Weight(stone): 20.0
```

```
Hamza's original weight:
```

```
Full Name: Hamza Bhatti, Age: 20, Year of Birth: 1993, Nationality: British, Height(meters): 1.77, Weight(stone): 11.0
```

```
Hamza's new weight
```

```
Full Name: Hamza Bhatti, Age: 20, Year of Birth: 1993, Nationality: British, Height(meters): 1.77, Weight(stone): 10.0
```

Searching

Another task was to search for an element. To allow the search to work, a fixed sized array with the Person ADT along with a linear search was used. The search cycled through the array and returned the value. An error does occur when an array is given but is not present. The error message that was implemented, shows an error.

Application

The application makes use of the previously created PersonADT. The objects are put into an array. The objects are printed and the method to search is called and returns the target value.

```
public class Searching
{
    /*
     * Using the Person ADT to put objects into an array and search it.
     */

    public static void main(String[] args)
    {
        // Variable for searching algorithm
        int left = 0;
        int right = 3;

        // Constructed an array which holds a max of 4 values
        PersonADT [] people = new PersonADT [4];

        // Populating array in certain indexes with objects using Person class
        people [0] = new PersonADT("Hamza Bhatti", 21, 1993, "British", 1.77,
11.00);

        people [1] = new PersonADT("John Doe", 28, 1988, "American", 1.80, 9.00);
        people [2] = new PersonADT("Homer Simpson", 30, 1970, "American", 1.90,
17);

        people [3] = new PersonADT("Peter Griffin", 30, 1970, "American", 1.90,
20);

        // Printing all the objects found in the array
        System.out.println("Printing all the objects in the array\n");
        for (int i = 0; i < people.length; i++)
            System.out.println(people[i]);

        System.out.println("");

        // Calling linear search algorithm
        System.out.println("Searching for Peter in index 3: ");
        int myMethod = linearSearch(people[3], people, left, right);

        // Printing return value from the linearSearch method

        if (myMethod != 0)
            System.out.println("Searched array is: " + people[myMethod]);
        else
            System.out.println("No such target");
    }

    // Linear Search algorithm
    static int linearSearch(Object target, Object[] a, int left,
        int right)
    {
        // Method to find an element in the array that matches the target

        // p will be the left most element of the array and will increment till
        it reaches the right
    }
}
```

```
        for (int p = left; p <= right; p ++)  
        {  
            // Using java method .equals to find the target element in the  
            array and return it  
            if (target.equals(a[p]))  
                return p;  
        }  
        // Otherwise return -1 if it can't be found  
        return 0;  
    }  
}
```

Execution

Printing all the objects in the array

Full Name: Hamza Bhatti, Age: 21, Year of Birth: 1993, Nationality: British, Height(meters): 1.77, Weight(stone): 11.0
Full Name: John Doe, Age: 28, Year of Birth: 1988, Nationality: American, Height(meters): 1.8, Weight(stone): 9.0
Full Name: Homer Simpson, Age: 30, Year of Birth: 1970, Nationality: American, Height(meters): 1.9, Weight(stone): 17.0
Full Name: Peter Griffin, Age: 30, Year of Birth: 1970, Nationality: American, Height(meters): 1.9, Weight(stone): 20.0

Searching for Peter in index 3:

Searched array is: Full Name: Peter Griffin, Age: 30, Year of Birth: 1970, Nationality: American, Height(meters): 1.9, Weight(stone): 20.0

Bracketing

This task required the use of stacks to implement a Bracketing algorithm. To use the stack operations, Linked list was imported into the program as it contained all the necessary methods to create a stack.

For this task, two different expressions were created that had brackets included in them. A method was called to read through the expression, find a bracket then add it to the stack. Once one bracket was found, another would be found. With another method the two brackets would be compared and identified as matching or not.

Application

//importing Linked List as it contains the methods needed to create and manipulate a stack

import java.util.LinkedList;

public class Bracketing

```
{
    /*
     * Program implement the bracket algorithm
     */
    public static void main(String[] args)
    {
        String mathExpression1;
        String mathExpression2;

        // String maths term well bracketed
        mathExpression1 = "s X (s - a) X (s - b) X (s - c)";

        // String maths term ill bracketed
        mathExpression2 = "s X (s - a) X (s - b X (s - c)";

        // Calling wellBracketed method for checking
        // expression 1 should be true, 2 false
        boolean bracketMethod1 = wellBracketed(mathExpression1);
        boolean bracketMethod2 = wellBracketed(mathExpression2);

        // Printing expressions
        System.out.println("Checking to see if expressions are"
            + " well bracketed:\n");

        // Printing expression 1 boolean
        System.out.println("Expression: " + mathExpression1 + "\n" +
            "Well Bracketed: " + bracketMethod1 + "\n");

        // Printing expression 2 boolean
        System.out.println("Expression: " + mathExpression2 + "\n" +
            "Well Bracketed: " + bracketMethod2 + "\n");
    }

    // Creating the wellBracketed method
    public static boolean wellBracketed (String phrase)
    {
        // This method tests to see if the string
        // is well bracketed

        // Creating a new stack/ Linked List
        LinkedList bracketStack = new LinkedList();

        // for loop starting from left to right of the string
        for (int i = 0; i < phrase.length(); i++)
        {
            // going through each character
            char sym = phrase.charAt(i);

            // store the bracket
            if (sym == '(' || sym == '[' || sym == '{')
                bracketStack.addLast(sym);
            else if (sym == ')' || sym == ']' || sym == '}')
            {

```

```

        if (bracketStack.isEmpty()) return false;
        char left = (char) bracketStack.removeLast();
        // checking if they both match with another method
        // if they dont return false
        if (! matched(left, sym)) return false;
    }
}
return (bracketStack.isEmpty());
}

// Method setting the conditions for each case
public static boolean matched (char left, char right)
{
    // return true if and only if Left and right are matching
    switch (left)
    {
        //
        case '(': return (right == ')');
        case '[': return (right == ']');
        case '{': return (right == '}');
        default: return false;
    }
}
}

```

Execution

Checking to see if expressions are well bracketted:

Expression: s X (s - a) X (s - b) X (s - c)
Well Bracketed: true

Expression: s X (s - a) X (s - b X (s - c)
Well Bracketed: false

Autotools

Autotools is used to build software. When a program is developed, Autotools can be used to compile the program. It allows the program to be installed onto the linux system for use across the computer and file system. Autotools is very useful for implementing files that are missing.

For this task, a student file was refactored into three different files. One file main.c which contained the main function, student.c which made use of the functions and finally a student.h file which contained the student structure and the function headers.

Along with the files a three files needed to be written. In the root folder of the project, makefile and configure files were created. Within the src folder, the student files and another makefile were included.

Configure.ac

This configure file is found in the root folder of the project.

```
# AC_INIT is the first macro, need to indicate the name of the...
# project and its version, can also provide email address
AC_INIT(Student, 0.1, foo@bar.com)

# Initialise environment for Automake, is always needed in...
# projects with Automake
AM_INIT_AUTOMAKE

# Identifying source directory where Makefile.am and code is found
AC_CONFIG_SRCDIR([src/])
AM_CONFIG_HEADER([src/config.h])

# Identifies the C preprocessor macro which allow identification...
# of header files
AC_DEFINE([PROGNAME],["Student"],["Student"])

# Checks for programs AND determines the C compiler to use
AC_PROG_CC

# Checks for header files
AC_HEADER_STDC

# Checks for typedefs, structures and compiler characteristics
AC_C_CONST
AC_TYPE_SIZE_T

# Checks for library functions
AC_FUNC_MALLOC

# Things to generate
AC_CONFIG_FILES([Makefile src/Makefile])

AC_OUTPUT
```

Makefile.am

This Makefile is found in the root folder of the project.

```
AUTOMAKE_OPTIONS = foreign
SUBDIRS = src
```


Main.c

This is one of three parts of the refactored code found in the src folder. This contains the main function.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "student.h"

int main()
{
    Student *stud1;

    stud1 = make_student(1001);
    make_student_active(stud1);
    set_student_name(stud1, "foobar");
    if (is_student_active(stud1)) {
        printf("student name is %s\n", stud1->name);
    }
    free_student(stud1);

    return EXIT_SUCCESS;
}
```

Student.c

This is the second part of the refactored code. It contains the use of the functions for the student.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "student.h"

Student *make_student(unsigned id) {
    Student *stud;

    if ((stud = (Student *)malloc(sizeof(Student))) == NULL) {
        printf("Failed to allocate Student structure!\n");
        exit(EXIT_FAILURE);
    }
    stud->active = 0;
    stud->id = id;
    stud->name = NULL;

    return stud;
}

void free_student(Student *stud) {
    free(stud->name);
    free(stud);
}

void make_student_active(Student *stud) {
    stud->active = 1;
}

void set_student_name(Student *stud, char *name)
{
    stud->name = strdup(name);
}

int is_student_active(Student *stud) {
    return stud->active;
}
```

Student.h

The final part of the refactored code which contains the student structure and the headings for the function.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    unsigned id;
    int active;
    char *name;
} Student;

Student *make_student(unsigned id);

void free_student(Student *stud);

void make_student_active(Student *stud);

void set_student_name(Student *stud, char *name);

int is_student_active(Student *stud);
```

Makefile.am 2

The second Makefile, found in the src folder.

```
INCLUDES = -Wall -g

bin_PROGRAMS = Student

Student_SOURCES = main.c student.c student.h config.h

Student_LDADD =
```

Compiling

When all the files were developed, a series of commands were then inputted into the terminal with the root folder of the project.

The first command was autoreconf – i which created the necessary files such as configure and other files that may have been missing. After this, the configure file was then run which checked to see if the system is able to compile the software and then generate Makefiles. Sudo make install was then entered which installs the software and creates an executable for the student files.

Root

```
hamza@hamza-X550CC:~/Hamza/ComputerScience4/Semester2_CS4/Laboratory/Autotools2$ ls
aclocal.m4      config.log      configure.ac    Makefile        missing
autom4te.cache  config.status   depcomp        Makefile.am     src
compile         configure       install-sh     Makefile.in     student-0.1.tar.gz
```

Src

```
hamza@hamza-X550CC:~/Hamza/ComputerScience4/Semester2_CS4/Laboratory/Autotools2/src$ ls
config.h  config.h.in  main.c  Makefile  Makefile.am  Makefile.in  stamp-h1  Student  student.c  student.h
```

Execution

```
hamza@hamza-X550CC:~/Hamza/ComputerScience4/Semester2_CS4/Laboratory/Autotools2/src$ ./Student
student name is foobar
```

Debugging

Ex1

This program is meant to allow the user to input an integer and print it. The program did not have a means to store the integer.

Solution

```
// The purpose of the program is to read an integer and print  
// it out. But the program seg faults. Use gdb to find the error.  
//  
// gcc -Wall -g -o ex1 ex1.c
```

```
// SOLVED  
#include <stdio.h>  
#include <stdlib.h>  
  
int main(int argc, char* argv[]){  
  
    // declaring variable  
    int x;  
  
    // prompting user for user to enter data  
    printf("Please enter an integer : ");  
  
    //Error here, expects %d to be stored. missing &x  
    scanf("%d",&x);  
  
    // printing the int stored in x  
    printf("the integer entered was %d \n", x);  
  
    return EXIT_SUCCESS;  
}
```

Execution

```
ugging/ex1$ ./main  
Please enter an integer : 3  
the integer entered was 3
```

Ex2

The file uses fopen to read a file. The original program had no file to open or while loop.

Solution

Ex2.c

```
// The purpose of the program is to read a set of strings from a
// file. But the program seg faults. Use gdb to find the error.
//
// gcc -Wall -g -o ex2 ex2.c

//SOLVED
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]){

    // ERROR here, trying to open file and read it.
    FILE* fp = fopen("text.txt", "r");
    // missing a value of array
    char* word[100]

    //wants to print the string text from the file
    while (fscanf(fp,"%s",word)>0)
    {
        printf("%s",word);
    }
    return 0;

}
```

Text.txt

Hello World!

Execution

```
ugging/ex2$ ./ex2
HelloWorld!hamza@hamza-X550CC:~/Hamza/ComputerScience4/Semester2_CS4/Lab
```

Ex3

This exercise wanted to print a value in an array. The value in the array was the maximum value that unix could write. The value n was changed

Solution

```
// The purpose of the program is to allocate an array of ints
// and initialize them to a random number. But the program seg
// faults. Use gdb to find the error.
//
// gcc -Wall -g -o ex2 ex2.c
```

```
//SOLVED
```

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
```

```
int main(int argc, char* argv[]){
```

```
    // this is printing the max value that ubuntu can print
    printf("%d \n", INT_MAX);
```

```
    // error here, if n is the max value
```

```
    int n = 5;
```

```
    // an array cannot show the maximum value, n should change
```

```
    int A[n];
```

```
    int i = 0;
```

```
    while (i<10)
```

```
    {
```

```
        A[i] = 12;
```

```
        printf("%d\n", A[i]);
```

```
        i++;
```

```
    }
```

```
    return EXIT_SUCCESS;
```

```
}
```

Execution

```
ugging/ex3$ ./ex3
2147483647
12
12
12
12
12
12
12
12
12
12
```

Ex4

For this task, a solution was not found, but a comment shows where the error is located.

Solution

*// This code may throw a segmentation fault. Use debugger to
// find out where the errors are and fix them.*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char* argv[]){

    // Error here
    char* name = malloc(strlen(argv[1]));

    // name is .....
    name = strcpy(name, argv[1]);

    //printing the name
    printf("%s \n", name);

    return EXIT_SUCCESS;
}
```

Ex5

A memory leak was found. As this is a XML doc, memory is allocated to the program. The memory was not freed up as it was commented out, which was then removed.

Solution

Ex5.c

```
// The purpose of this program is to navigate an XML tree
// and print element names. Use valgrind to find the memory
// leak and fix it.

/**
 * section: Tree
 * synopsis: Navigates a tree to print element names
 * purpose: Parse a file to a tree, use xmlDocGetRootElement() to
 *          get the root element, then walk the document and print
 *          all the element name in document order.
 * usage: tree1 filename_or_URL
 * test: tree1 test2.xml > tree1.tmp && diff tree1.tmp $(srcdir)/tree1.res
 * author: Dodji Seketeli
 * copy: see Copyright for the status of this software.
 */
#include <stdio.h>
#include <libxml/parser.h>
#include <libxml/tree.h>

// Valgind says there is a memory leak!!

/*
 * To compile this file using gcc you can type
 * gcc -Wall -g -o ex5 ex5.c `xml2-config --cflags --libs`
 * echo "<hello>Student</hello>" > file.xml
 * ./ex5 file.xml
 */
/* print_element_names:
 * @a_node: the initial xml node to consider.
 *
 * Prints the names of the all the xml elements
 * that are siblings or children of a given xml node.
 */
static void print_element_names(xmlNode * a_node)
{
    xmlNode *cur_node = NULL;

    for (cur_node = a_node; cur_node; cur_node = cur_node->next) {
        if (cur_node->type == XML_ELEMENT_NODE) {
            printf("node type: Element, name: %s\n", cur_node->name);
        }

        print_element_names(cur_node->children);
    }
}

/**
 * Simple example to parse a file called "file.xml",
 * walk down the DOM, and print the name of the
 * xml elements nodes.
 */
int
main(int argc, char **argv)
{
```

```
xmlDoc *doc = NULL;
xmlNode *root_element = NULL;

if (argc != 2)
    return(1);

/*
 * this initialize the library and check potential ABI mismatches
 * between the version it was compiled for and the actual shared
 * library used.
 */
LIBXML_TEST_VERSION

/*parse the file and get the DOM */
doc = xmlReadFile(argv[1], NULL, 0);

if (doc == NULL) {
    printf("error: could not parse file %s\n", argv[1]);
}

/*Get the root element node */
root_element = xmlDocGetRootElement(doc);

print_element_names(root_element);

// ERROR HERE! xmlFreeDoc was commented out!!
/*free the document */
xmlFreeDoc(doc);

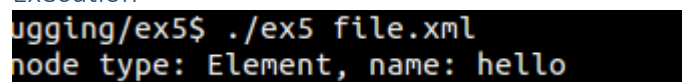
/*
 *Free the global variables that may
 *have been allocated by the parser.
 */
xmlCleanupParser();

return 0;
}
```

File.xml

```
<hello>Student</hello>
```

Execution



```
g++/ex5$ ./ex5 file.xml
node type: Element, name: hello
```


Ex6

A memory leak was found within the program. The xmlFreeDoc was in the wrong place.

Solution

```
// This program shows how to create and XML DOM using Libxml2.
//
// Scenario: the program was originally written by senior
// programmer Alyssa P. Hacker. It was then maintained by
// the company junior programmer Louis Reasoner which recently
// started using vi. The program was working by now it segfaults.
//
// Use both gdb and valgrind to find the fix the error.

/*
 * section: Tree
 * synopsis: Creates a tree
 * purpose: Shows how to create document, nodes and dump it to stdout or file.
 * usage:   tree2 <filename> -Default output: stdout
 * test:    tree2 > tree2.tmp && diff tree2.tmp $(srcdir)/tree2.res
 * author:   Lucas Brasilino <brasilino@recife.pe.gov.br>
 * copy:     see Copyright for the status of this software
 */

// SOLVED!!
#include <stdio.h>
#include <libxml/parser.h>
#include <libxml/tree.h>

/*
 * gcc -Wall -g -o ex6 ex6.c `xml2-config --cflags --libs`
 * ./ex6
 */

/* A simple example how to create DOM. Libxml2 automagically
 * allocates the necessary amount of memory to it.
 */
int
main(int argc, char **argv)
{
    xmlDocPtr doc = NULL;          /* document pointer */
    xmlNodePtr root_node = NULL, node = NULL, node1 = NULL; /* node pointers */
    xmlDtdPtr dtd = NULL;          /* DTD pointer */
    char buff[256];
    int i, j;

    LIBXML_TEST_VERSION;

    /*
     * Creates a new document, a node and set it as a root node
     */
    doc = xmlNewDoc(BAD_CAST "1.0");
    root_node = xmlNewNode(NULL, BAD_CAST "root");
    xmlDocSetRootElement(doc, root_node);

    /*
     * Creates a DTD declaration. Isn't mandatory.
     */
    dtd = xmlCreateIntSubset(doc, BAD_CAST "root", NULL, BAD_CAST "tree2.dtd");

    /*
     * xmlNewChild() creates a new node, which is "attached" as child node
     * of root_node node.
     */
}
```

```

xmlNewChild(root_node, NULL, BAD_CAST "node1",
            BAD_CAST "content of node 1");
/*
 * The same as above, but the new child node doesn't have a content
 */
xmlNewChild(root_node, NULL, BAD_CAST "node2", NULL);

/*
 * xmlNewProp() creates attributes, which is "attached" to an node.
 * It returns xmlAttrPtr, which isn't used here.
 */
node =
    xmlNewChild(root_node, NULL, BAD_CAST "node3",
                BAD_CAST "this node has attributes");
xmlNewProp(node, BAD_CAST "attribute", BAD_CAST "yes");
xmlNewProp(node, BAD_CAST "foo", BAD_CAST "bar");

/*
 * Here goes another way to create nodes. xmlNewNode() and xmlNewText
 * creates a node and a text node separately. They are "attached"
 * by xmlAddChild()
 */
node = xmlNewNode(NULL, BAD_CAST "node4");
node1 = xmlNewText(BAD_CAST
                  "other way to create content (which is also a node)");
xmlAddChild(node, node1);
xmlAddChild(root_node, node);

/*
 * A simple loop that "automates" nodes creation
 */
for (i = 5; i < 7; i++) {
    sprintf(buff, "node%d", i);
    node = xmlNewChild(root_node, NULL, BAD_CAST buff, NULL);
    for (j = 1; j < 4; j++) {
        sprintf(buff, "node%d%d", i, j);
        node1 = xmlNewChild(node, NULL, BAD_CAST buff, NULL);
        xmlNewProp(node1, BAD_CAST "odd", BAD_CAST((j % 2) ? "no" : "yes"));
    }
}

/*
 * Dumping document to stdio or file
 */
xmlSaveFormatFileEnc(argc > 1 ? argv[1] : "-", doc, "UTF-8", 1);

// ERROR was in wrong position
/*free the document */
xmlFreeDoc(doc);

/*
 *Free the global variables that may
 *have been allocated by the parser.
 */
xmlCleanupParser();

/*
 * this is to debug memory for regression tests
 */
xmlMemoryDump();
return(0);
}

```

Execution

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root SYSTEM "tree2.dtd">
<root>
  <node1>content of node 1</node1>
  <node2/>
  <node3 attribute="yes" foo="bar">this node has attributes</node3>
  <node4>other way to create content (which is also a node)</node4>
  <node5>
    <node51 odd="no"/>
    <node52 odd="yes"/>
    <node53 odd="no"/>
  </node5>
  <node6>
    <node61 odd="no"/>
    <node62 odd="yes"/>
    <node63 odd="no"/>
  </node6>
</root>
```