CURTIN UNIVERSITY
FACULTY OF SCIENCE AND ENGINEERING: SCHOOL OF ELECTRICAL ENGINEERING, COMPUTING AND MATH SCIENCE

Programming Languages Assignment

Student name: *Tawana Kwaramba: 19476700*

Course: *Programming Languages - COMP2007* – Lecturer: *Ascsociate Lecturer: Arlen Brower*
Due date: *October 25th, 2021*

Contents

List of Figures

List of Tables

1. Fortran

```
      program fizzBuzz
      integer times, ii

      write (*,*) 'Enter amount of times you want to run fizz buzz: '
      read (*,*) times
      write (*,*) 'times running fizz buzz algorithm: ', times
      ii = 0

      do while (ii < times)
      ii = ii + 1

*starting off with the number 15 as that is going to be the biggest
   number,
*and it's going to short circuit the lower numbers. 15 is going to be
   divisible
*with less things than 3 and 5
      if (mod(ii,15) == 0) then
       print *, 'fizz buzz'

      else if (mod(ii,5) == 0) then
       print  *, 'buzz'

      else if (mod(ii,3) == 0) then
       print *, 'fizz'

      else
        print *, ii
      endif

      end do

      stop
      end
```

Figure 1: FizzBuzz: Fortran Code

2. Algo68

```
BEGIN
    print (("Enter the amount of times which you want to run the
       programme: "));
    INT times = read int;

    print (("Running fizz  buzz times of ", times));
    print ((" ", new line));

    FOR ii FROM 1 TO times DO
        IF ((ii MOD 15) = 0) THEN
            print(("Fizz buzz", new line))

        ELIF ((ii MOD 5) = 0) THEN
            print(("buzz", new line))

        ELIF ((ii MOD 3) = 0) THEN
            print(("Fizz", new line))
        ELSE
            print((ii, new line))
        FI
    OD
END
```

Figure 2: FizzBuzz: Algo68 Code

3. ADA

```
with Ada.Text_IO;
use Ada.Text_IO;


procedure BubbleSort is
    --  defining an unconstrained array of integers
    --type arrayOfIntegers is Array  (Natural range <>) of Integer;
    type arrayOfIntegers is Array (0 .. 10) of Integer;
    elements : arrayOfIntegers := (15, 14, 13, 12, 11, 10, 9, 8, 7, 6,
       5);

    procedure bubbleSort (arrayToSort : in out arrayOfIntegers) is
        -- the value which is going to be used a place holder when we'
           re going to
        -- be switching the elements of an array around
        temp : Integer;

        begin
            for ii in reverse arrayToSort'Range loop
                for jj in arrayToSort'First .. ii loop
                    if arrayToSort(ii) < arrayToSort(jj) then
                        -- actually swapping the elements around in the
                            array
                        temp := arrayToSort(jj);
                        arrayToSort(jj) := arrayToSort(ii);
                        arrayToSort(ii) := temp;
                    end if;
                end loop;
            end loop;
        end bubblesort;

    procedure printArrayValues(sortedArray : in out arrayOfIntegers) is
        begin
            for ii in sortedArray'Range  loop
                Put(Integer'Image(sortedArray(ii)));
            end loop;
            Put_Line(" ");
        end printArrayValues;


begin
    -- I am treating this a main method where yo're actually going to
       run the
    -- programme
    Put_Line("Before bubble sort: ");
    printArrayValues(elements);

    bubbleSort(elements);
    Put_Line("After the bubble sort: ");
    printArrayValues(elements);
end BubbleSort;
```

Figure 3: FizzBuzz: ADA Code

4. Scripting Languages

```perl
#!usr/bin/perl
use warnings;
use File::Find;

# recursively searching all the files found in my home directory, change s
#search path to home machine

my $searchPath = "/home/19476700/";
my $fileFind = ".conf";
find(\&filesWanted, $searchPath);



sub filesWanted {
    #print "$File::Find::name\n if $File::Find::name =~ /Prac01/";
    my $currDirectory = $File::Find::name;
    #print "$File::Find::name\n" if  index(File::Find::name $fileFind) != -1;
    print "$currDirectory\n" if (index($currDirectory, $fileFind) != -1)
}
```

Figure 4: Finding Config files: Perl Code

4.1. Perl.

```ruby
#!/usr/bin/env ruby
require 'find'
#making sure that I only access my student id, change to the home directory of
#personnal machine
#outContents =  Dir["/home/19476700/**/*.conf*"]
#puts outContents

Find.find("/home/19476700/") { |f| puts f if  f.include? ".conf"}
```

Figure 5: Finding Config files: Ruby Code

4.2. Ruby.

4.3. Bash.

```bash
#!/bin/bash

#saving the current directory so we can switch back it after we have found
#the files which we need to find
currDirectory="$(pwd)"
saveLocation="$currDirectory"/out_bashOne.file""
saveLocationTwo="$currDirectory"/out_bashTwo.file""
cd ~/
find . -print | grep .conf  > "$saveLocation"
find . -type f -name "*.conf*" > "$saveLocationTwo"
cd $currDirectory

echo "First search for anything with .conf"
cat "$saveLocation"
echo "Second search for file names specifically with .conf in it"
cat "$saveLocationTwo"
```

Figure 6: Finding Config files: Bash Code

5. Small Talk

```
"hello world programm"
"so you will use the \\ operation to find the remainder of a number"

'Hello World' printNl !



1 to: 100 do: [ :ii |
    "Will need to start at the highest comparison number so the lower numbers
    don't short circuit the algorithm"
    (ii \\ 15 == 0) ifTrue: [ 'Fizz Buzz' printNl ] ifFalse:
    [ (ii \\ 5 == 0) ifTrue: ['Buzz' printNl] ifFalse:
    [ (ii \\ 3 == 0) ifTrue: ['Fizz' printNl] ifFalse:
    [ ii printNl ] ] ]
] !
```

Figure 7: FizzBuzz: Smalltalk Code

6. C++

You should split this, code into two parts for both the classes and the actual quick sort code

```
class Book
{
    private:
        int bookID;
        std::string bookName;
        std::string ISBN;
        bool validateIntegerInput(int);
        bool validateName(std::string);

    public:
        Book();
        Book(int, std::string, std::string);
        int getBookID();
        std::string getBookName();
        std::string getISBN();

        void setBookID(int);
        void setBookName(std::string);
        void setBookISBN(std::string);
        ~Book();
};
```

Figure 8: Book: C++ Book Header Code

```cpp
#include <string>
#include <cstdlib>
#include <iostream>
#include "Book.h"

//default constructors
Book::Book()
{
    bookID   = 1;
    bookName = "Discourses and Selected - Epictectus";
    ISBN = "12345";
}

//alternate constructors
Book::Book(int inBookID, std::string inBookName, std::string inISBN)
{
    bookID = inBookID;
    bookName = inBookName;
    ISBN = inISBN;
}

//defining that the behaviours of the class is going to be in
int Book::getBookID()
{
    return bookID;
}

std::string  Book::getBookName()
{
    return bookName;
}

std::string Book::getISBN()
{
    return ISBN;
}

void Book::setBookID(int inID)
{
    if(validateIntegerInput(inID))
    {
        bookID = inID;
    }
}

void Book::setBookName(std::string inName)
{
    if(validateName(inName))
    {
        bookName = inName;
    }
}

void Book::setBookISBN(std::string inBookISBN)
{
    if(validateName(inBookISBN))
```

```
1 void printElements(Book* inBooks[], int inSize);
2 void quickSort(Book* inBooks[], int inSize);
3 void swap(Book* bookOne, Book* bookTwo);
4 int doPartition(Book* inBooks[], int leftIndx, int rightIndx);
```

Figure 10: Book: C++ Quick Sort Header Code

```
#include <iostream>
#include <string>
#include <cstdlib>
#include "Book.h"
#include "quickSort.h"

int main()
{
    int size = 10;
    std::string placeHolder = "NOT IN USE";
    //creating some objects so we can perform a quick sort on them
    Book* book1 = new Book();
    Book* book2 = new Book();
    Book* book3 = new Book();
    Book* book4 = new Book(102800, "Seven habits of highly effecitve
       people -"
    " Stephen R. Covey", placeHolder);
    Book* book5 = new Book(21000, "Poor Charlie's Almanack: The wit and
        "
    "wisdom of Charles T", placeHolder);
    Book* book6 = new Book(999,  "Sam Walton - Sam Walton", placeHolder
       );
    Book* book7 = new Book(10, "Tao of Munger - David Clark",
       placeHolder);
    Book* book8 = new Book(2, "Principles - Ray Dailo", placeHolder);
    Book* book9 = new Book(5, "Meditations - Marcus Aurelius",
       placeHolder);
    Book* book10 = new Book(10, "Man's search Meaning - Victor E. "
    " Frankl", placeHolder);

    Book* bookCollection[size] = {
    book5,
    book6,
    book7,
    book8,
    book9,
    book10,
    book1,
    book2,
    book3,
    book4,
    };

    //int num = elements[0] -> getBookID();

    std::cout << "Sorting books by Book ID, using quick sort" << std::
       endl;

    std::cout << "The original books collection" <<std::endl;

    printElements(bookCollection, size);

    std::cout << "testing out the swap function" << std::endl;
    swap(bookCollection[1], bookCollection[0]);
    printElements(bookCollection, size);
```

```
#File Created: Friday 9th of August 10:14
#AUTHOR: Tawana Kwaramba
#LAST MODIFIED:
#MODIFIED BY:
#PURPOSE: A make file to run all the download relateded programmes

# Makefile Variables
CC = g++
EXEC = quickSort
OBJ = Book.o quickSort.o

$(EXEC) : $(OBJ)
	$(CC) $(OBJ) -o $(EXEC)

Book.o : Book.cpp Book.h
	$(CC) $(CFLAGS) -c Book.cpp

quickSort.o : quickSort.cpp quickSort.h
	$(CC) $(CFLAGS) -c quickSort.cpp

clean:
	rm -f $(EXEC) $(TEST) $(OBJ) $(TEST_OBJ) $(gameTest) $(game_OBJ
	    )
```

Figure 12: QuickSort: Makefile

7. Prolog

8. Scheme