CodeLab
Bonus Exercises & Mini Briefs

Creative Computing
The School of Creative Industries
Bath Spa University

Using this workbook

The following workbook should be used if you are looking for some tougher challenges. You should ensure you have worked through the main CodeLab workbook and are comfortable with all the techniques introduced before using this workbook.

These exercises included here can be solved in a variety of ways. Tackle them in any order you wish. Each will require a mixture of the techniques introduced through the workshops so far (selection, loops, arrays, functions etc). As new techniques are introduced you might think of ways to improve your original solutions, so go back and iterate on your solutions accordingly. For example could previous solutions benefit from Object Oriented Programming (OOP) techniques, or enhanced by implementing a openFrameworks based graphical user interface (GUI)? These techniques are introduced in CodeLab II?

As there are a variety of possible ways to complete these exercises only some solutions have been provided at the end of the workbook.

Some of these exercises, particularly the later ones in the list will take a number of hours / days to complete and are there to help push the boundaries of your C++ knowledge.

Warm Up Exercises

Getting Loopy

Write a program that will print the following pattern:

```
1******
12*****
123****
1234***
12345**
123456*
1234567
```

Function Error

Write a function that asks the user to enter a number that is greater than 0. The function will keep on asking the user for the number until it is valid. The function will then return the number to the main body of the program.

Even More Function Errors

Write a function that asks the user to respond by 'Y', 'y', 'yes', 'YES' or 'N', 'n', 'no', 'NO'. The function keeps on asking until the user enters the correct information. The function will return True if the user entered a form of Yes, and False otherwise.

Hint: converting string via toupper or tolower might streamline your code

---

Who ate all the pies

Five football fans go to a game and as per tradition the all scoff down some pies. Write a program that asks the user to enter the number of pies eaten by each person. Next output the following information:

- Who ate what amount of pies
- Who ate the most pies
- Who ate the least pies
- Who ate what amount of pies in the correct order

Extension Problem:

Rather than having the user input the data implement file handling to read the data from a file.(See here for information on file handling: https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm)

The Tax Problem

Write a program the works out the tax due for company employees. The program should use the following information to determine the tax due.

- If annual gross pay is less than the tax threshold then: *Annual tax due = £0*
- If annual gross pay is more than the tax threshold then: *Annual tax due = (annual gross pay – tax threshold) * 0.2*
- *Tax threshold = 10 * tax code*

Think through logic of problem

- Need to input each pair of values in turn and for each pair calculate tax due
- Therefore need a repetition where each pair of values is input and processed before going to the next pair.

Possible pseudo-code scheme for solving the problem

```
Input two data values (call them weeklypay and taxcode)
While weeklypay and taxcode are not both zero{
      set threshold to 10 * taxcode
     if 52 * weeklypay < threshold{
            output that no tax has to be paid
      }else{
            set taxdue to (52*weeklypay-threshold) * 0.2 / by 52)
            output the value of the tax due
      }
      input two more data values
}
```

The program should implement functions.

Test data

	Weekly pay	Tax code	Output
1	100	1000	No tax due
2	500	1000	???
3	300	800	???
4	0	0	Thank you for using the tax calculator code by 'Your name'

Larger Programs

Magic 8 Ball

Write a magic 8 ball program. You should allow the user to input a question, to which the program will provide a cryptic response. E.g:

Q: Shall I go out tonight?
A: Outlook looks good

Extension Problem:

Rather than hardcoding the 8 ball responses into the program implement file handling to read the responses from a file. (See here for information on file handling: https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm)

Number Guessing Game

Create a number guessing game where the user must guess what the number is between 1-100. The game should allow the user to keep guessing until they get it right. After each guess the game should provide feedback informing the user how far away their guess is.

Extension Problems:

- Implement a scoring system based on the number of guesses the user takes to get the answer right
- Rank this scoring system so the user gets a grade or stars
  - e.g. 1-3 guesses = A, 5-7 guesses = B etc..
- Set the original number randomly
- Implement different modes (e.g. easy and hard), For example in easy mode the game provides feedback on how far away the guess is, whilst in hard mode no feedback is provided.

Quiz

Create a quiz program with at least 10 questions. The choice of questions are up to you (e.g. Capital Cities, Football Teams, Music Artists). The following are a list of desirable features:

- Scoring system
- Record of right and wrong answers
- Random ordering of questions on each play
- Replay functionality without exiting the program

Rock, Paper, Scissors

Write a program to play rock, paper, scissors against the computer. The game should be best of five goes.

Extension problem

- Revise the program so the player has the option of playing against the computer or against a second player.
- Can you enhance this program by implementing a GUI (e.g: OpenFrameworks)

Tic Tac Toe

Write a program to play tic, tac, toe (naughts and crosses) against the computer. The game should be best of five goes.

Extension problem

- Revise the program so the player has the option of playing against the computer or against a second player.
- Can you enhance this program by implementing a GUI (e.g: OpenFrameworks)

Tamagotchi

Write a program for a virtual pet caring game. Suggested features for this include:

- Allow user to name the pet
- Allow user to pick type of pet
- Allow user to feed the pet
- Allow user to play with the pet
- If the user neglects the pet they could fall ill and die
- If the user overfeeds the pet they could fall ill and die

Extension Problem

- Turn the program into a Virtual Zoo by enabling multiple pets at once
- Ensure the program makes use of functions and OOP techniques
- Can you enhance this program by implementing a GUI (e.g: SDL / OpenFrameworks)

Maths Quiz

This program will present the user with quiz of arithmetic problems. Each "play" of the quiz will be 10 questions. The user should initially be presented with a short menu of options on difficulty level. It could look something like this:

```
DIFFICULTY LEVEL

1. Easy
2. Moderate
3. Advanced
```

The difficulty levels determine the number of digits in the operands to be added or subtracted. Easy means only single digit numbers; moderate means double digit numbers; and advanced means 4-digit numbers. After the user picks the level they desire, your program presents problems that look like this:

```
45 + 9 =    _
34 - 88 =   _
etc
```

But for each problem presented, the user is given a chance to answer. If the answer is correct, another problem is presented. If the answer is wrong, the user is to be given one more chance at that problem. Once your program has moved on to the next problem, the correctness/incorrectness of the preceding problem is tallied. The number of correct and incorrect answers is to be presented at the termination of the quiz along with percentage correct.

So, how are the problems produced? Well, you will use the random number generator provided by the compiler to determine:

1. the values to be added or subtracted
2. whether the problem is addition or subtraction

Hint: Take a look at the handy hints code file on AURA for how to implement random numbers.

Here is a listing of the functions required in your program. These functions should make use of parameters and return values as appropriate. You may include others if you see fit.

- a displayMenu function that displays the difficulty level at the beginning.
- a randomInt function that determines the values used in each question and the min and max values of the ints should be based on the difficulty level chosen.
- a decideOperation function that randomly decides whether the problem is an addition or subtraction problem and returns a char.
- a displayProblem function that displays the question to the user and accepts their answer.
- a isCorrect function that checks whether the users answer was correct.
- a displayMessage function that outputs "correct" or "incorrect" dependent on the value returned by the isCorrectFunction
- a displayFinalResults function that outputs the number of correct and the number of

wrong answers.

Once the user has finished the quiz, prompt them to see if they'd like to do it again.

Extension Problems:

- Include a function that will choose at random from a list of 4 output messages for good answers and a list of 4 output messages for wrong answers.
- Make the number of tries at each problem a prompted for parameter for each run of the quiz at the beginning of the quiz.
- Provide a ranking system based on the users score (e.g. 10/10 = A)
- Add more complex questions (e.g. include multiplication & division, or increase the potential number of digits used in the problem.
- Organise your code across different files. (e.g. one for the main program, a header file for function declarations and any global variables, a cpp file for the function definitions
- Set a time limit for answering the questions (e.g 10 seconds). If the player exceeds this limit the game is over. *Hint: start_time and end_time from #include <ctime> may be useful here.*
- Can you enhance this program by implementing a GUI (e.g: OpenFrameworks)

WarBots

Concept: In the far future, humanity uses robots to settle wars. These are called warbots. You are to build and test a program to help build the ultimate warbot.

Each bot has the following characteristics.

- Name
- Aim: which is the percentage chance of hitting between 0- 60%
- Strength: Which is the range of damage that it inflicts if it hits. This is between 1-10 points
- Armour: between 0-60%, this is the chance of a successful hit being deflected and causing no damage.
- Hits to kill: is the number of hits to kill the warbot.
- Dexterity: no maximum

The programme must allow the values of each warbot to be entered via the keyboard. Each robot has 20 points to allocate. The rules for building are:

- Aim: costs 1 construction point per 10% (maximum of 60%)
- Strength: 1 point per strength point (1-10 points)
- Armour: costs 1 construction point per 10% (maximum of 60%)
- Hits to kill: 1 construction point per 2 hits
- Dexerity: costs 1 construction point per point (no maximum)

The two warbots fight. The fight should have the following actions

- The one with the highest dexterity attacks first, if the dexterity is the same, then it is random which one attacks first.
- The attacking robot should determine if it hits by generating a random number e.g. if the aim is 50%, then it hits 50% of the time.
- The defending robot the rolls to see if its armour deflects the attack, leading to no damage.
- If the defending robot hits, deduct the strength of the attacking robot from the hits to kill of the defender.
- The defending robot now attacks.
- This carries on until the one of the robots is killed.

Extension problem

- Ensure the program makes use of functions and OOP techniques
- Can you enhance this program by implementing a GUI (e.g: OpenFrameworks)

Professor Scott and the Temple of Doom

A text based adventure game.

The party led by well-known explorer Professor Scott (and his faithful companions) is facing his most complex challenge yet. He is attempting to retrieve the Crystal Skulls for the infamous Temple of Doom. Obviously, the Temple is protected by monsters and traps. The party should start outside the Temple.

Each time someone is killed, one of the faithful companions should be removed first. Professor Scott is the last one to be killed and if he dies, then the adventure is over.

All probabilities should be shown on the screen e.g:

“You enter the cave and there is a spike trap. It stands 40% of working." You roll 39%. Oh dear, 1 of your party is killed. What do you wish to do next?”

Game Rules

Each room entered should state the name of the room and any items/ monsters/ traps in it.

Player commands include N (move north), S (move south), W (move west) E (move east), A (attack), F (flee), P item name (e.g. P torch), I (inventory- a list of the what the party is carrying).

The temple should consist of a maximum of 30 rooms/ corridors/ other locations. Items can include treasure, torch, weapon etc…

Any room with a trap only activates once (i.e. if you return to the room, nothing happens).

A monster(s) can attack anyone in the room until the monster is dead or the party flees. The chance of trap working or a monster successfully attacking (and killing one of the party) should be random e.g. a pit trap stands a 40% chance of killing someone if the part does not have a torch, 20% if they do.

Extension Problem

- The contents of each room should be read from a file. The player should have the option to save the game during play
- The program should implement functions and OOP techniques
- Can you enhance this program by implementing a GUI (e.g: OpenFrameworks)

Some Solutions

Getting Loopy

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

int main()
{
   for(int i   1; i <  7; i++){
       for(int j   1; j <  7; j++){
           if(j< i)
               cout << j;
           else
               cout << "*";
       }
       cout << endl;
   }
   cin.get();
   return 0;
}
```

Function Error

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

int numberEntry(){
   int x;
   do{
       cout << "Enter a number greater than 0:" << endl;
       cin >> x;
   }while(x < 1);
   return x;
}
int main()
{
   int x   numberEntry();
   cout << "Your number is: " << x << endl;
   cin.get();
   return 0;
}
```

Even More Function Errors

```
#include<iostream>
#include<string>
using namespace std;

bool textEntry(){
   string str;
   //do while loop asks for input
   do{
       cout << "Enter Y / y / yes / YES / N / n / No / NO: " << endl;
       cin >> str;
       for(int i   0; i < str.length(); i++){
           str[i]   tolower(str[i]);
       }
   }while(str! "y" && str! "yes" && str! "n" && str! "no");

   //if statement returns appropriate true or false
   if(str  "yes" || str  "y"){
       return true;
   }else{
       return false;
   }

}
int main()
{
   //call to text entry function and assign return value to bool
variable
   bool x   textEntry();

   //output statement using ternary operator to streamline if/else
   cout << (x ? "You entered Yes" : "You entered No") << endl;

   /*IF STATEMENT ALTERNATIVE TO TERNARY
   if(x==true){
       cout << "You entered Yes" << endl;
   }else{
       cout << "You entered No" << endl;
   }
   */
   cin.get();
   return 0;
}
```

Who ate all the pies

```
#include<string>
using namespace std;
void sort(int pies[], string fans[], int size);

int main(){
    int numPies[5];
    string fans[5];
    for(int i   0; i < 5; i++){
        cout << "Enter Football Fan name: " << endl;
        getline(cin, fans[i]);
        cout << "Enter Number of Pies they ate" << endl;
        cin >> numPies[i];
        cin.ignore(1000, '\n');
    }
    int max   0;
    int least   0;
    for(int x   0; x < 5; x++){
        cout << fans[x] << " ate " << numPies[x] << " Pies" << endl;
    }
    for(int j   0; j < 5; j++){
        if (numPies[j] > numPies[max])
            max   j;
    }
    for(int k   0; k < 5; k++){
        if (numPies[k] < numPies[least])
            least   k;
    }
    cout << "\n========================\n" << endl;
    cout << fans[max] << " ate the most Pies (" << numPies[max] << ")"
<< endl;
    cout << fans[least] << " ate the least Pies (" << numPies[least] <<
")" <<
    endl;
    sort(numPies, fans, 5); // call to sort function passing in arrays
    cout << "\n========================\n" << endl;
    for(int x   0; x < 5; x++){
        cout << fans[x] << " ate " << numPies[x] << " Pies" << endl;
    }
    cin.get();
    return 0;
}
void sort(int pies[], string fans[], int size) {
    int a, b, tempP;
    string tempF;
    for (a   0; a < size   1; a++){
        for (b   a+1; b < size; b++) {
            //swap both pies and fans arrays
            //so they remain in corresponding order
            if (pies[a] > pies[b]) {
                tempP   pies[b];
                tempF   fans[b];
                pies[b]   pies[a];
                fans[b]   fans[a];
                pies[a]   tempP;
                fans[a]   tempF;
            }
        }
    }
}
```

The Tax Problem

```
#include<iostream>
using namespace std;
/* Declaration of the functions*/
int inputTaxCode();
int inputWeeklyPay();
void taxDue (int WeeklyPay,int TaxCode);
int main()
{
    int weeklyPay   inputWeeklyPay(); /*call function InputPay */
    if(weeklyPay !  -1){
        do /*start outer loop*/
        {
            int taxCode   inputTaxCode(); /*call function InputTaxcode
*/
            if (weeklyPay > 0 && taxCode > 0){ /*check if valid values
have been entered*/
                taxDue(weeklyPay, taxCode); /*call function Taxdue*/
            }
            else{
                /*error message*/
                cout<<"This information can not be correct, please try
again.";
            }
            weeklyPay   inputWeeklyPay();
        } while (weeklyPay !  -1); /*end of outer loop. keeps program
cycling
                                     until
                                     a value of 0 is entered as weekly
pay*/
    }
    cout << "End of Program";
    cin.get();
    return 0;
}
//****************************************
int inputWeeklyPay()
{
    int weeklyPay;
    cout << "Please enter weekly pay (-1 to quit): "; //output message
to screen
    cin >> weeklyPay; //input value entered to integer WeeklyPay
    return (weeklyPay);
}
```

```
//****************************************
int inputTaxCode()
{
    int taxCode;
    cout << "Please enter taxcode: "; //output to screen
    cin >> taxCode; //input value entered as taxcode
    return (taxCode);
}
//**************************************************
void taxDue(int weeklyPay, int taxCode) /*open function*/
{
    int taxThreshold   taxCode * 10; //set Threshold value to 10 * the
inputed taxcode
    if (52 * weeklyPay < taxThreshold)
    {
        cout << "No tax has to be paid." << endl; //print to screen
    }
    else
    {
        /*calculate the taxdue by the week*/
        double taxDue   ((((52 * weeklyPay)   taxThreshold) * 0.2) /
52);
        /*output message and taxdue to screen*/
        cout<<"\nTax due this week for this case is: " << taxDue <<
endl;
    }
}
```

Magic 8 Ball

```
#include <iostream>
//these headers useful for random numbers
#include <stdlib.h>
#include <time.h>
using namespace std;

string returnResponse(); //declare function that will return cryptic response

int main(){
    srand(time(NULL)); //set random seed to get random numbers each time
    cout << "Welcome to the magic 8 ball" << endl; //output header
    cout << "----------------------------" << endl;
    string question; //string to hold question
    char confirm; //char to ask user if they want to continue
    do{ //using do while as want to get input at least once
        cout << "Ask me a question: " << endl; //ask user for input
        getline(cin, question); //get user question
        cout << returnResponse() << endl; //invoke response function and output
returned string
        cout << "Continue (Y)?" << endl; //ask user if they want to continue
        cin >> confirm; //get input
        cin.ignore(1000, '\n'); //ignore input to force stream onto next line to
get subsequent question input
    }while(confirm!  'Y' || confirm !  'y'); //loop continues while user enters
Y
    return 0;
}
string returnResponse(){
    int r   rand() % 19; //get random number between 0 and 19
    //array full of responses
    string responseList[]   {"As I see it, yes", "Ask again later", "Better not
tell you now", "Cannot predict now",
        "Concentrate and ask again", "Don't count on it", "It is certain", "It
is decidedly so", "Most likely",
        "My reply is no" "My sources say no", "Outlook good", "Outlook not so
good", "Reply hazy try again",
        "Signs point to yes", "Very doubtful", "Without a doubt", "Yes", "Yes,
definitely", "You may rely on it"
    };
    return responseList[r]; //return response using random number as index
}
```

Number Guessing Game

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

int main() {
    cout << "Guess the number (1-100):" << endl; //output information to user
    int answer   66; //set the value of the answer
    int userAnswer; //variable to store user input
    cin >> userAnswer; //get user input

    if(answer  userAnswer){ //check for correct answer
        cout << "Correct Answer!!" << endl;
    }else if(answer>userAnswer){ //if not correct check if answer is more than
userAnswer to avoid negative results in subtraction
        cout << "Wrong Answer!! You were out by: " << answer userAnswer  <<
endl;
    }else{
        cout << "Wrong Answer!! You were out by: " << userAnswer answer  <<
endl;
    }
}
```

Quiz

```
#include <iostream>
using namespace std;
int main() {
    //set question strings
    string question[]
{"France","England","Wales","Spain","Germany","Scotland","Ireland","Italy",
"Sweden", "Denmark"};
    //set answer strings in lower case for easy handling later
    string answer[]
{"paris","london","cardiff","madrid","berlin","edinburgh","dublin","rome",
"stockholm","copenhagen"};

    int correctCount   0; //variable to track correct answers
    string userAnswer; //variable for user answer

    //output quiz header
    cout << "Welcome to the Capital Quiz!!\n";
    cout << "===============================\n" << endl;

    //use a for loop to run through each of the questions
    for(int i   0; i < 10; i++){
        //the format for each question in this example quiz is the same so I can
easily access the question string from the array and add it to the desired
format
        cout << "Q" << i+1 << ": What is the capital of " << question[i] << "?"
<< endl;

        cin >> userAnswer; //get the users answer

        //this for loop converts answer to lower case to handle different
formats the user might have answered in
        for(int j   0; j < userAnswer.length(); j++){
            userAnswer[j]   tolower(userAnswer[j]);
        }
        //compare the user answer with the answer and provide feedback
        if(userAnswer    answer[i]){
            correctCount++;
            cout << "Correct, well done!\n" << endl;
        }else{
            cout << "Unlucky, that's incorrect\n" << endl;
        }
    }

    char grade; //grade variable

    //assign grade based on amount correct answers
    if(correctCount >  0 && correctCount < 5){
        grade   'C';
    }else if(correctCount > 4 && correctCount < 8){
        grade   'B';
    }else{
        grade   'A';
    }
    //output final results
    cout << "\n============================================\n";
    cout << "You got " << correctCount << " out of 10 correct.\n";
    cout << "Your grade is: " << grade << "\n";
    cout << "Thank you for taking the quiz" << "\n";
    cout << "============================================\n" << endl;
    return 0;
}
```