
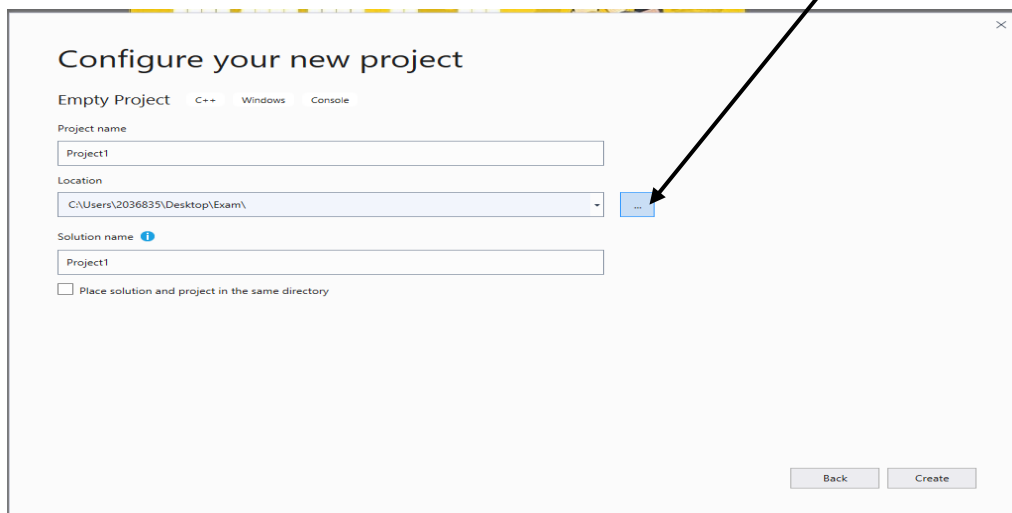


CSC2040 Assessed Practical 3

Wednesday 11th December 2019

Before you begin this test please read carefully and follow instructions. As you write programs for the test, saving as you go, the following procedure will ensure that if your computer crashes or if you upload the incorrect files your work will not be lost and can be restored.

- 1 Log on to the machine in the usual way.
- 2 Using File Explorer  locate the S:\ drive (public_shares (\\nas02.eecs.qub.c.uk)).
- 3 In the folder CSC2040 Exam 1920 you will find an application:
1 Create_Exam_Folder
- 4 Run Create_Exam_Folder by double clicking on it and then clicking Run. An exam folder appears on the desktop. Press any key to exit the blue screen.
- 5 Open Visual Studio and create a new Empty Project. By clicking this button, change the Location of the project to the Exam folder on the desktop. See an example below (your location will be slightly different but should end \Desktop\Exam)



Name the Solution Test3 and the first program Project1.

For the questions below, look for the following five files in the Canvas. The files includes the various library classes needed in your programs:

StackNode.h, LinkedStack.h
ListNode.h, LinkedList.h
TreeNode.h

Project 1 Linked stack (30 marks)

Inside Project1 create a new source file Project1.cpp. In Project1.cpp,

(1) Write a function

```
void randNumStack(LinkedStack<int>& s, int num, int upper)
```

which takes an existing (empty) `int` stack `s`, and pushes a sequence of `num` random numbers within the range `0~upper` (inclusive) on to this stack.

(2) Below the randNumStack function, write another function

```
int countNumStack(LinkedStack<int>& s, int lower, int upper)
```

This function takes an existing `int` stack `s` and scans all its numbers to count how many numbers fall within the given bounds `lower~upper` (inclusive). It finally returns the count. Note, after the function call the supplied stack should be left as it was. You should accomplish the operations by using only stacks. Do not edit the supplied library classes.

(3) For test, below the two functions write a main function, which first defines an empty `int` stack `s`, next calls `randNumStack` to setup `s` with `num = 1000` and `upper = 100`, and finally calls `countNumStack` to print out the counts with appropriate bounds (e.g. your program should print out 1000 when called with `lower = 0` and `upper = 100`).

Project 2 Linked list (35 marks)

In Solution Test3 add a new project Project2. Inside Project2 create a new source file Project2.cpp to contain the following programs.

(1) Building a list to store arrays of items

Write a template function

```
void array2list(T* array, int num_items, LinkedList<T>& list, int list_p)
```

This function takes a type-T array, `array`, containing `num_items` items, and inserts these items one after another into a given list, `list`, starting from location `list_p`.

(2) Print items in a list

Write a template function

```
void printList(LinkedList<T>& list, int list_p, int num_items)
```

This function takes a list and prints out `num_items` items from location `list_p`. The operation should not change the list. Based on the given library class you may accomplish this operation by using two methods: (a) call function `getAt`, and (b) call function `getNext`. In `printList` write code to print the list using both methods, and clearly label the method which you think is faster for large `num_items`.

Write a main function to test your programs properly.

Project 3 Trees (35 marks)

In Solution Test3 add a new project Project3. In this project create a new source file Project3.cpp for the following programs.

(1) Write a function

```
TreeNode<int>* BST()
```

This function uses dynamic allocation to construct a sorted binary tree (i.e. a binary search tree - BST) for the random number sequence: 321, 511, 41, 168, 814, 623, 409. It returns the pointer to the tree.

(2) Write a function

```
bool searchBST(TreeNode<int>* tree, int key)
```

This function conducts search within a given binary search tree (BST), `tree`, for the given item, `key`. It returns true for finding the `key` and false for not. Your code should make good use of the BST structure to minimise the number of comparisons. To grant access to the private members of the library class you can make `searchBST` to be a friend function inside the class by including the following statement:

```
friend bool searchBST(TreeNode<int>* tree, int key);
```

(3) Writing a main function to include suitable code to test your functions, for example,

```
TreeNode<int>* tree = BST();
cout << searchBST(tree, 41) << endl;
cout << searchBST(tree, 168) << endl;
cout << searchBST(tree, 321) << endl;
cout << searchBST(tree, 409) << endl;
cout << searchBST(tree, 511) << endl;
cout << searchBST(tree, 623) << endl;
cout << searchBST(tree, 814) << endl;
cout << searchBST(tree, 6) << endl;
cout << searchBST(tree, 900) << endl;
.....
```

Submission Notes

1. Save your project.
2. Make sure you have the following **four** source files ready for submission:
 - Project1.cpp
 - Project2.cpp
 - Project3.cpp, TreeNode.h
3. Locate your Solution folder Test3 containing all your source files. ZIP this folder (by right clicking the folder and choosing 'Send To' and selecting Compressed (Zipped) folder). Rename the zip file to XXXXXXXX.zip, where XXXXXXXX is your student number.
4. Check that your .zip file contains your source files (double click on the .zip file. You do not need to extract the files again).
5. In case something goes wrong with your submission, go to the folder CSC2040 Exam 1920 and find the application:
 - 2 Backup_Exam_Folder

Double click on it. You can make a backup of your zip file on to a USB drive, and/or email it to yourself.
6. On the course's Assignments page for Practical Test 3 on Canvas choose Submit Assignment, followed by File Upload. Click Choose File and locate the zip file you have just created. Click Submit Assignment again.