


University of West London School of Computing and Engineering	 UNIVERSITY OF WEST LONDON
---	--

Title	Coursework			
Module	Algorithms and Data Types			
Module Code	CP40064E			
Module Leader:	Sama Aleshaiker			
Set by:	Sama Aleshaiker			
Moderated by:	Hafiz Sherazi			
Assignment:	Coursework			
Hand in arrangements:	Online submission via Blackboard			
	Element	Type	Weighting	Due Date
	1	Coursework	40%	Sunday 15 th May 2021 23:59
	Extensions will only be granted in exceptional circumstances. Documentary evidence will be required. Extensions must be agreed before the deadline. A student who fails to submit course work or dissertation by the applicable deadline shall be deemed to have failed to submit to assessment.			
Learning outcomes:	1- Apply mathematical skills to approximate running time of programs by analysing and interpreting algorithms 2- Design algorithms using iteration and recursion 3- Use and implement Lists, Stack and Queues 4- Understand and employ Trees and ADT 5- Describe and use a variety of sorting algorithms			

- For this coursework, you should create a logbook containing your answers to the questions below. You can find a template for the logbook in the Assessments section of the module on Blackboard.
- Coursework should be all printed and submitted either word or PFD document. Photos or scanning to handwritten work will not be accepted and marked.
- Any screenshot of your code **must** include comment that state your name and student ID within the code such as:

```
1 # student_firstname_surname STUDENT ID: 21xxxxxx ADT coursework
```

And any screenshot of your console output **must** include your name and student ID:

```
student_firstname_surname STUDENT ID:21xxxxxx ADT coursework
```

Section 1 – Algorithm Efficiency

Question 1

What is the time complexity of the following three algorithms? Express your answer in terms of Big-O notation and justify your answer:

A)

```
def display_names_with_ID(names):
    ID = 0;
    for n in names:
        print("ID for ", n, "is", ID)
        ID = ID + 1
```

[2 marks]

B)

```
def find_item(mylist, item):
    if len(mylist) == 0:
        return False
    else:
        n = len(mylist)//2
        print("Midpoint =", mylist[n])
        if mylist[n] == item:
            return True
        else:
            if item < mylist[n]:
                return find_item(mylist[:n], item)
            else:
                return find_item(mylist[n+1:], item)
```

[2 marks]

C)

```
print("Times tables grid")
n = 10
for i in range(1, n+1):
    print(i, end="\t")
for i in range(0, 75):
    print("=", end='')
print()
for i in range(1, n+1):
    for j in range(1, n+1):
        print(i * j, end="\t")
```

[2 marks]

Section 2 – Recursion

Question 2

Write a Python function that returns the nth Fibonacci number, where n is an integer passed as a parameter to the function.

[6 marks]

Section 3 – Stacks

Question 3

Evaluate the following postfix expressions, giving your answer as a single number:

- a) $5\ 4\ * \ 9\ +$
- b) $8\ 3\ 7\ * \ + \ 4\ -$
- c) $16\ 12\ + \ 4\ * \ 2\ /$

[3 marks]

Convert the following infix expressions to postfix:

- d) $9 + 2 * 20 - 4$
- e) $4 + 5 * 7 / 3$
- f) $(20 - 8) * (42 - 16) / (21 + 6)$

[3 marks]

Section 4 – Queues

Question 4

First implement a Stack to insert your student ID digit by digit in order such as 213xxxxx then empty your stack and insert the output into the Queue. What will be the output if you empty the Queue? Demonstrate your work by showing the steps for both Stack and Queue and screenshot of your code and output. Assuming both stack and queue are initially empty.

[8 marks]

Section 5 – Linked Lists

Question 5

Write a Python program that creates an unordered, singly-linked list consisting of 8 items. Each item in the linked list should be a number. You can use the code given in the lecture slides for your Node and UnorderedList classes.

Your node class should contain the following functions:

- a constructor
- `get_data()` – returns the data in the node
- `get_next()` – returns the next node in the list
- `set_data()` – sets the data in the node to the value given as a parameter
- `set_next()` – sets the node that this node links to, to the node given as a parameter

Your UnorderedList class should contain the following functions:

- a constructor
- `add()` – adds a node to the head of the list, containing the number given in the parameter
- `is_empty()` – returns True if the list is empty, and False otherwise
- `size()` – returns the size of the list
- `print_list()` – displays the contents of all nodes in the list

Add code to create an UnorderedList object, and call its `add()` function to add 8 items to the list. Call the `is_empty()`, `size()` and `print_list()` functions to show that they work. In your logbook, show your code and the output when you run the code.

The 8 numbers you will add them to the list should be your student ID.

Example output:

```
Creating a linked list containing the following numbers: 21, 35, 40, 50, 15, 8
Displaying the contents of the list using the print_list() function
8 15 50 40 35 21
Testing the is_empty() function
False
Testing the size() function
6
```

[5 marks]

Add a function called `search()` to your UnorderedList class in Question 5.1. This function should take a number as a parameter and return True if the number is contained in the linked list, or False if the number is not in the list.

Add code to your program to allow the user to enter a number, and search for the number in your linked list. Display a message to the user indicating if the number they entered was found in the list or not. In your logbook, include your code, and also output that shows you have tested the search function for a number in the list and a number not in the list.

Example output:

```
Please enter a number to search for in the list: 50
True
Please enter a number to search for in the list: 28
False
```

[3 marks]

Section 6 – Sorting

Question 6

Selection sort: use your student ID as list of integers to be sorted by a selection sort algorithm. Show the contents of the list after each pass through the list.

The list should be your 8-digit student ID.

Example: If your student ID is 21234567 then your list will be

[2, 1, 2, 3, 4, 5, 6, 7]

[3 marks]

Insertion sort: use your student ID as list of integers to be sorted by a insertion sort algorithm. Show the contents of the list after each pass through the list.

The list should be your 8-digit student ID.

Example: If your student ID is 21234567 then your list will be

[2, 1, 2, 3, 4, 5, 6, 7]

[3 marks]

Marking Criteria			
	Weak / 0% – 39%	Sufficient / 40% - 69% G	Good / 70% - 100%
1 6 marks	Wrong answer	Incorrect Big O notation/ Justification not given	Correct Big O notation with justification
2 6 marks	Incorrect function	Partially implemented code/ your code and output doesn't include your name and student ID	Fully implemented function with screenshots of the code and output
3 6 marks	1 mark for each correct answer		
4 8 marks	Incorrect function	Partially implemented code/ your code and output doesn't include your name and student ID	Fully implemented function with screenshots of the code and output
5 8 marks	Incorrect output	Partially implemented code/ your code and output doesn't include your name and student ID	Fully implemented code with screenshots of the output
6 6 marks	Incorrect sorting	Partially correct sorting algorithm/ your student ID wasn't used	Correct sorting algorithm step by step