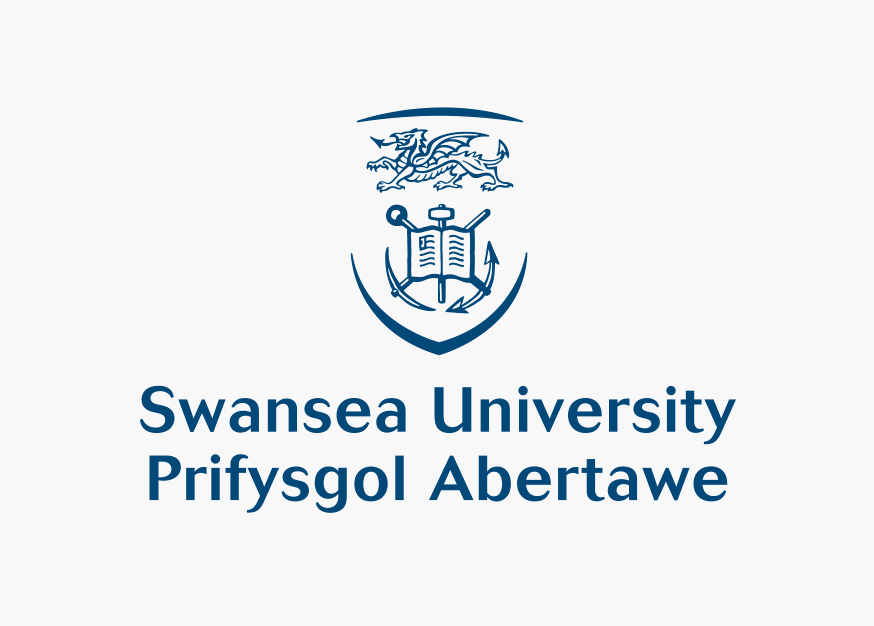
* cslogotrans*

**CS-155 Concepts of Computer Science II 2017/2018**

**Assessed Coursework 1**

**Aims:**

1. To ensure you understand the material discussed in lectures.
2. To give you practice in following the various algorithms discussed in the lectures.

***Fill in the form fields as appropriate, save and submit via Blackboard.***

***The first ten questions are multiple choice questions. In each case there is only one correct answer for which 1 mark will be awarded. Incorrect or missing answers will score 0.***

1. Which of the following tracks the progress of a program during execution?

A) process management

B) memory management

C) multiprogramming

D) timesharing

E) CPU scheduling

1. Which of the following is a technique for keeping more than one process in memory at the same time?

A) process management

B) memory management

C) multiprogramming

D) timesharing

E) CPU scheduling

1. Which of the following shares CPU time among multiple processes to create the illusion that each user has a dedicated machine?

A) process management

B) memory management

C) multiprogramming

D) timesharing

E) CPU scheduling

1. Which of the following describes a memory management technique in which a program is divided into fixed sized sections and stored into areas of memory called frames?

A) single contiguous

B) logical address

C) physical address

D) round robin

E) paged

1. Which of the following describes the act of bringing in a page from secondary memory, which may cause another to be written back to secondary memory?

A) swapping

B) context switch

C) demand paging

D) thrashing

E) virtual memory

1. Which of the following describes the situation in which memory appears to be unlimited because the program size is not restricted?

A) swapping

B) context switch

C) demand paging

D) thrashing

E) virtual memory

1. To which state does the currently executing process return when it is interrupted by the operating system?

A) ready

B) new

C) blocked

D) suspended

E) running

1. In which state does a process reside if it is not currently resident in main memory?

A) ready

B) new

C) blocked

D) suspended

E) running

1. In which state does a process reside if it does not have a needed resource, such as a page from secondary memory?

A) ready

B) new

C) blocked

D) suspended

E) running

1. To which state does a process move when the CPU scheduling algorithm determines it may next use the CPU?

A) ready

B) new

C) blocked

D) suspended

E) running

1. Consider the following Binary Search Algorithm

*upper = length - 1;*

*lower = 0;*

*while (upper >= lower)*

*{ mid\_pt = (upper + lower) / 2;*

*if* ***(data[mid\_pt] < target) ; comparison A***

*{ lower = mid\_pt + 1; }*

*else* ***if (data[mid\_pt] == target) ; comparison B***

*{ return mid\_pt; }*

*else { upper = mid\_pt – 1;}*

*}*

*//target not found*

Give the values for lower, upper and mid-pt for each iteration of the while loop and state how many times will the two comparisons A and B be executed using this algorithm to find the value 77 in the following array? You may not need to complete all the rows.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] | [12] |
| 14 | 27 | 29 | 38 | 42 | 55 | 57 | 61 | 64 | 69 | 77 | 79 | 84 |

|  |  |  |
| --- | --- | --- |
| Lower | Upper | Mid-pt |
| 0 | 12 | 6 |
| 7 | 12 | 9 |
| 10 | 12 | 11 |
| 10 | 10 | 10 |
|  |  |  |

Comparison A executed 2 times; Comparison B executed 1 times;

***[10 marks]***

1. A Bubble Sort algorithm as discussed in lectures is given below

*index1 = 0;*

*repeat exchange = false;*

*{ for index2 ← length – 1 downto index1*

*{ if data[index2] < data[index2-1]*

*{ //****exchange***

*exchange = true;*

*tmp = data[index2];*

*data[index2] = data[index2-1];*

*data[index2-1] = tmp;*

*}*

*}*

*index1 = index1 + 1;*

*} until (not exchange)*

Apply this algorithm to the following data. Give the contents of the array after each pass (repeat loop) is completed. For each pass how many exchanges are made?

*Note it may not be necessary to use all 7 passes.*

Original Data 14 27 12 56 63 72 8 10

After Pass 1 14 12 27 56 63 8 70 72

Exchanges 3

After Pass 2 12 14 27 56 8 10 65 72

Exchanges 3

After Pass 3 12 14 27 8 10 56 65 72

Exchanges 2

After Pass 4 12 14 8 10 27 56 63 72

Exchanges 2

After Pass 5 12 8 10 14 27 56 63 72

Exchanges 2

After Pass 6 8 10 12 14 27 56 63 72

Exchanges 2

After Pass 7 8 10 12 14 27 56 63 72

Exchanges 0

How many comparisons will be made in total? 49

***[10 marks]***

1. Consider the following quicksort algorithm

*Quicksort(first, last)*

*{IF (first < last) // There is more than one item*

*splitVal = data[first];*

*splitPoint = Split(splitVal);*

*left = first + 1;*

*right = last;*

*WHILE (left <= right)*

*{Increment left until data[left] > splitVal OR left > right;*

*Decrement right until data[right] < splitVal OR left > right;*

*IF(left < right)*

*Swap data[left] and data[right]*

*}*

*splitPoint = right;*

*Swap data[first] and data[splitPoint];*

*Quicksort(first, splitPoint - 1);*

*Quicksort(splitPoint + 1, last)*

}

Each recursive call to the Quicksort function acts on a portion of the array to be sorted. Illustrate how the algorithm works on the same data as in the previous questions by completing the trace below. For each call of Quicksort give the index values for first and last as well as the value (splitVal) used as a pivot value. Also show the state of the array being sorted after the evaluation of all the Quicksort functions applied to the array.

**You may not need to complete all the tables below.**

Original Data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
| 14 | 27 | 12 | 56 | 63 | 72 | 8 | 10 |

|  |  |  |
| --- | --- | --- |
| First | Last | splitVal |
| 0 | 7 | 14 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
| 14 | 10 | 12 | 8 | 63 | 72 | 56 | 27 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First | Last | splitVal |  | First | Last | splitVal |
| 0 | 3 | 8 |  | 4 | 7 | 63 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
| 8 | 10 | 12 | 14 | 27 | 56 | 72 | 63 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |
| 0 | 1 | 8 |  | 2 | 3 | 12 |  | 4 | 5 | 27 |  | 6 | 7 | 72 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
| 8 | 10 | 12 | 14 | 27 | 56 | 63 | 72 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |  | First | Last | splitVal |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|  |  |  |  |  |  |  |  |

***[20 marks]***

1. Complete the following table. Where appropriate you may use scientific notation (*eg 1.5E+6)* Give values to 6 significant figures.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N** | **log2n** | **log10n** | **n2** | **2n** | **nlog2n** |
| **1** | 0 | 0 | 1 | 2 | 0 |
| **5** | 2.32193 | 0.698970 | 25 | 32 | 11.6096 |
| **10** | 3.32193 | 1 | 100 | 1024 | 33.2193 |
| **50** | 5.64386 | 1.69897 | 2500 | 1.12590E+15 | 282.193 |
| **100** | 6.64386 | 2 | 10000 | 1.26765E+30 | 664.386 |
| **500** | 8.96578 | 2.69897 | 250000 | 3.27339E+150 | 4482.89 |
| **1000** | 9.96578 | 3 | 1000000 | 1.07151E+301 | 9965.78 |

***[25 marks]***

1. Complete the following table stating the start and end times for each process using the first-come, first-served algorithm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P1 | P2 | P3 | P4 | P5 |
| Service time | 40 | 70 | 100 | 20 | 80 |
| Arrival Time | 0 | 20 | 40 | 60 | 80 |
| Start Time | 0 | 40 | 110 | 210 | 230 |
| End Time | 40 | 110 | 210 | 230 | 310 |

***[5 marks]***

1. Complete the following table stating the start and end times for each process using the shortest-job-next algorithm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P1 | P2 | P3 | P4 | P5 |
| Service time | 40 | 70 | 100 | 20 | 80 |
| Arrival Time | 0 | 20 | 40 | 60 | 80 |
| Start Time | 0 | 40 | 210 | 110 | 130 |
| End Time | 40 | 110 | 310 | 130 | 210 |

***[5 marks]***

1. Complete the following table stating the start, pause, restart and end times for each process (including any times when a process is swapped out) using the round robin algorithm and a time slice of 30. You may not need to complete all entries.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P1 | P2 | P3 | P4 | P5 |
| Service time | 40 | 70 | 100 | 20 | 80 |
| Arrival Time | 0 | 20 | 40 | 60 | 80 |
| Start Time | 0 | 30 | 60 | 90 | 110 |
| Pause | 30 | 60 | 90 | 110 | 140 |
| Restart | 140 | 150 | 180 |  | 210 |
| Pause | 150 | 180 | 210 |  | 240 |
| Restart |  | 240 | 250 |  | 280 |
| Pause |  | 250 | 280 |  | 300 |
| Restart |  |  | 300 |  |  |
| End Time |  |  | 310 |  |  |

***[15 marks]***

**Deadline:- 11am Wednesday 28th February 2018 (via Blackboard).**

***Only doc(x) forms will be accepted.***