CS-155

Concepts of Computer Science 1

2015/16

Assessed Coursework 1

Learning Outcomes:

To ensure you understand the material discussed in lectures and 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 is performed by an operating system to keep track of which programs are in memory and where they are?

A) process management ☐

B) memory management ■

C) multiprogramming ☐

D) timesharing ☐

E) CPU scheduling ☐

2. Which of the following is the process of deciding which program is to be executed and in which order?

A) process management ☐

B) memory management ☐

C) multiprogramming ☐

D) timesharing ☐

E) CPU scheduling ■

3. Which of the following describes how an operating system keeps track of the progress of a program if it is interrupted?

A) process management ☐

B) memory management ☐

C) multiprogramming ■

D) timesharing ☐

E) CPU scheduling ☐

4. Which of the following describes a reference to a stored value relative to the program making the reference?

A) single contiguous ☐

B) logical address ■

C) physical address ☐

D) round robin ☐

E) paged ☐

5. Which of the following describes the allocation of memory according to the unique demands of a program?

A) swapping ☐

B) fixed-partition ☐

C) dynamic-partition ■

D) thrashing ☐

E) virtual memory ☐

6. Which of the following takes advantage of the fact that it is not necessary for all of a program to be in memory at the same time?

A) swapping ☐

B) context switch ☐

C) demand paging ■

D) thrashing ☐

E) virtual memory ☐

7. In which state does a process reside if there are no barriers to its execution?

A) ready ■

B) new ☐

C) blocked ☐

D) suspended ☐

E) running ☐

8. In which state does a process reside if it is currently being executed?

A) ready ☐

B) new ☐

C) blocked ☐

D) suspended ☐

E) running ■

9. In which state does a process move to if it is removed from the CPU before it has finished?

A) ready ☐

B) new ☐

C) blocked ■

D) suspended ☐

E) running ☐

10. To which state does a process move when it has finished its work?

A) ready ■

B) new ☐

C) blocked ☐

D) terminated ☐

E) running ☐

11. Consider the following Binary Search Algorithm

upper = length -1;

lower = 1;

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 88 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] |
| 12 | 26 | 30 | 38 | 46 | 52 | 60 | 68 | 75 | 82 | 88 | 92 | 97 |

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

Comparison A executed \_\_\_2\_\_times; Comparison B executed \_\_\_1\_\_times; **[10 marks]**

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

index1 = 1;

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 12 24 15 38 17 32 54 21

After Pass 1 12 15 24 17 32 38 21 54

Exchanges 5

After Pass 2 12 15 17 24 32 21 38 54

Exchanges 2

After Pass 3 12 15 17 24 21 32 38 54

Exchanges 1

After Pass 4 12 15 17 21 24 32 38 54

Exchanges 1

After Pass 5 12 15 17 21 24 32 38 54

Exchanges 0

After Pass 6 --complete--

Exchanges 0

How many comparisons will be made in total? 34

**[10 marks]**

13. 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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
| 12 | 24 | 15 | 38 | 17 | 32 | 54 | 21 |

|  |  |  |
| --- | --- | --- |
| **First** | **Last** | **SplitVal** |
| **0** | **7** | **12** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|  |  |  |  |  |  |  |  |
| **First** | **Last** | **SplitVal** |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **First** | **Last** | **SplitVal** |
|  |  |  |
|  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|  |  |  |  |  |  |  |  |
| **First** | **Last** | **SplitVal** |
|  |  |  |
|  |  |  |
| **First** | **Last** | **SplitVal** |
|  |  |  |
|  |  |  |
| **First** | **Last** | **SplitVal** |
|  |  |  |
|  |  |  |
| **First** | **Last** | **SplitVal** |
|  |  |  |
|  |  |  |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First | Last | SplitVal | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | | | | | | | | | First | | | | | | Last | | | | | SplitVal | | | |  | | | | | | | | |  | | | | | |  | | | | |  | | | |  | | | | | | First | | | | | | Last | | | | | | SplitVal | | | | |  | | | | | |  | | | | | |  | | | | | |  | | | | |  | | | | First | | | | | | Last | | | | | | SplitVal | | | | | |  | | | |  | | | | | |  | | | | | |  | | | | | |  | | First | | | | | Last | | | | | | SplitVal | | | | | | |  | |  | | | | |  | | | | | |  | | | | | | |  | First | | | | Last | | | | | | SplitVal | | | | | | |  |  | | | |  | | | | | |  | | | | | | | 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]**

14. 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** | **0.00000** | **0.00000** | **1** | **2** | **0** |
| **10** | **3.32193** | **1.00000** | **100** | **20** | **33.2198** |
| **100** | **6.64386** | **2.00000** | **10000** | **200** | **664.385** |
| **250** | **7.96578** | **2.39794** | **62500** | **500** | **1991.45** |
| **500** | **8.96578** | **2.69897** | **250000** | **1000** | **4482.89** |
| **750** | **9.55075** | **2.87506** | **562500** | **1500** | **7163.06** |
| **1000** | **9.96578** | **3.00000** | **1E+6** | **2000** | **9965.78** |

**[25 marks]**

15. 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 | 30 | 80 | 110 | 30 | 90 |
| Arrival Time | 0 | 20 | 40 | 60 | 80 |
| Start time | 0 | 50 | 170 | 340 | 450 |
| End time | 30 | 130 | 280 | 370 | 540 |

[5 marks]

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P1 | P2 | P3 | P4 | P5 |
| Service Time | 40 | 70 | 100 | 20 | 80 |
| Arrival Time | 0 | 20 | 40 | 60 | 80 |
| Start time | 0 | 140 | 250 | 100 | 430 |
| End time | 40 | 210 | 350 | 120 | 510 |

**[5 marks]**

17. 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 25.

You may not need to complete all entries.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P1 | P2 | P3 | P4 | P5 |
| Service time | 30 | 75 | 110 | 40 | 75 |
| Arrival time | 0 | 20 | 40 | 60 | 80 |
| Start time | 0 | 25 | 50 | 75 | 100 |
| Pause | 25 | 50 | 75 | 100 | 125 |
| Restart | 125 | 130 | 155 | 180 | 195 |
| Pause | 130 | 155 | 180 | 195 | 220 |
| Restart | - | 220 | 245 | - | 270 |
| Pause | - | 245 | 270 | - | 295 |
| Restart | - | - | 295 | - | - |
| End time | - | - | 320 | - | - |

[15 marks]

**Deadline Thursday 2nd March 2017 11:00 a.m.**

**You MUST use the electronic form on Blackboard, fill it in electronically and submit it to Blackboard.**