Programming Fundamentals

Question 1

(a) Using an example, explain a situation where you need to use the Heap.

[2 marks]

Solution: - when we want to define dynamic variables or - when we do not know the size requirements for an array

(b) What are the advantages of separating the interface of an abstract data type (ADT) from its implementation?

[3 marks]

Solution: Then programmers do not need to worry about the implementation (1 mark) but just know that it works. Unnecessary information is hidden from people who don't need to know it (1 mark) which allows you to carry out internal changes without your users having to rewrite your code (1mark).

(c) Write a code segment that will search an array of integers, `myArray`, to determine whether a given integer is located in an odd array index. Assume that the integer is stored in a variable called `targetItem`. If the item is found in an odd array index (for example: odd array indices are: 1,3,5 etc.), the message "Found" is displayed together with the array index of the item. Otherwise the message "Item not found" is displayed. You may assume that `myArray` and `targetItem` are both declared and initialised.

[7 marks

Solution: correct loop conditions - one condition to cover the length of the array and the other to include some kind of flag to stop the loop (2 marks) loop should skip values (2 marks) - some other less solution (using mod operator) - 1 value correct conditions within the loop (1 mark) - comparing value At the end of the loop for displaying the correct messages (2 marks)

(d) The following code fragment shows the creation of an $n \times n$ identity matrix, where the matrix is dynamically allocated and initialised as a one-dimensional `int` array (assume `n` is known and defined).

```
int *eye = new int[n*n];
for (i=0;i<n;i++)
{
   for (j=0;j<n;j++)
   {
      if (i==j)
```

```
            eye[i*n+j] = 1;
        else
            eye[i*n+j] = 0;
    }
}
```

Rewrite the code fragment such that the identity matrix is dynamically allocated and initialised as a two-dimensional `int` *array, i.e., elements in the matrix can be accessed by* `eye[i][j]`.

Solution:

```
int **eye = new int*[n];  // 2 marks
for (i=0;i<n;i++)
{
    eye[i] = new int[n];  // 2 marks
    for (j=0;j<n;j++)
    {
        if (i==j)
            eye[i][j] = 1;  // 1 mark
        else
            eye[i][j] = 0;  // 1 mark
    }
}
```

[Total for Question 1

Inheritance and Object Oriented Programming

Question 2

(a) i. Using examples explain
- overloading
- redefining
- overriding

within the context of C++

[6 marks]

Solution: (1 mark for explanation and 1 mark for the example). Students can also use one example. Overloading is where a single function name has two or more signatures with different parameter lists void f(int a); void f(int a, int b); Overriding is where a child class redeclares a function from its parent that uses the virtual keyword, the program chooses which function to call at run-time.

ii. Explain how concepts of overloading, redefining and overriding affects code-maintenance.

[2 marks]

Solution: code-maintenance is made easier because these concepts allow code-reuse.

(b) Consider the following two classes.

```
class Pet
{
public:
    void print();  // Prints name of pet.
    string name;
};
class Dog : public Pet
{
    void print();  // Prints name and breed of dog.
    string breed;
};
```

i. What problem occurs when the following statements are executed? Note: the following operations are all legal.

```
Dog vdog;
Pet vpet;
vdog.name = "Scooby Doo";
vdog.breed = "Great Dane";
vpet = vdog;
```

[2 marks]

Solution: slicing problem

ii. Consider a different code fragment shown in the following.

```
Dog vdog;
Pet vpet;
vpet.name = "Marmaduke";
vdog = vpet;
```

Is the operation corresponding to the last statement legal? Briefly explain.

[2 marks]

Solution: Not legal. The compiler will not know how to assign all member variable values in the derived class object.

(c) Write a class for an object called `Job` that has two integer variables `startingTime` and `endingTime`. `Job` should also have a default constructor and two public functions `getCost` and `reschedule` with the signatures:

```
int calculateCost(int hourlyRate);
void reschedule(int newStartingTime, int newEndingTime);
```

Function `calculateCost` returns the cost of the job. First it needs to find the processing time, which is calculated as the difference between `endingTime` and `startingTime`. Then this difference is multiplied by the hourly rate to obtain the cost.

Function `reschedule` changes the values of `startingTime` and `endingTime`, respectively, to `newStartingTime` and `newEndingTime` when the new ending time is greater than or equal to the new starting time. Otherwise, no action is performed.

The default constructor initialises both `startingTime` and `endingTime` to 0.

Assuming that the C++ header code has been declared separately, write the C++ *body* code for these functions and the default constructor.

Solution:

```
Job::Job()
{
  startingTime=0;
  endingTime=0;
}
  // OR Job::Job() : startingTime(0), endingTime(0) {}
  // 2 marks for correct signature and assignment
```

```
  // 1 mark for adding the class in all the functions

int Job::calculateCost(in t hourlyRate)
{
  return (hourlyRate *(endingTime-startingTime));
}
  // 1 mark for correct implementation

void Job::reschedule(int newStartingTime, int newEndingTime)
{
  if (newEndingTime<newStartingTime) return;
  // the condition may be implemented differently
  startingTime=newStartingTime;
  endingTime=newEndingTime;
  // 2 marks for correct implementation and
  // for no return or for retuning nothing
  // (both are valid for void).
}
```

[6 marks]

[Total for Question 2: 18 marks]

ırsion

stion 3

(a) Using an example, explain the role of the base condition in a recursive function?

[3 marks

Solution: terminating the recursion. (1 mark)
example (2 marks)

(b) Write a c++ function for computing x^n for a positive integer n and a real number x.

[5 mark

Solution: 1 mark for the base case, 3 marks for recursive call. 1 mark for correct function signature.

i. Change the function to use tail recursion. If your function is already tail recursive, write 'Already Tail Recursive' in your answer book.

[3 ma

Solution: The modified version should have used tail recursion. 3 marks for recursive call. Tail recursion works by ensuring the last operation of a recursive function is a call to a recursive function, which allows the compiler to finish the first call before starting the second (1 mark). This prevents over-use of the stack potentially avoiding stack overflows (1 mark).

ii. *Explain how tail recursion* improves the efficiency of the recursion?

[3 n

Solution: Tail recursion works by ensuring the last operation of a recursive function is a call to a recursive function, which allows the compiler to finish the first call before starting the second. only needs to store the return location not the state/variables associated with the call (1 mark). This prevents over-use of the stack potentially avoiding stack overflows (1 mark).

[Total for Question 3: 14

tion 4

(a) What is the definition of $f(n) = O(g(n))$? Give an example of a function that is in $O(n^2)$ and formally prove that your function is in $O(n^2)$.

[5

Solution: There exist constants n_0 and C, so that for all $n >$ or $\geq n_0$, we have $f(n) \leq Cg(n)$. (2 marks) example:f(n)=n or anything else that is correct (1 mark) formal proof: find correct c and n_0 such that inequality holds for all $n >$ or $\geq n_0$. 2 marks for any correct pair of constants.

(b) Given that $f(n) \in \Omega(n^3)$ and $g(n) \in O(n^2)$, can we always say that $f(n) \times g(n) \in O(n^5)$? Prove your answer.

[

Solution: No. (2 mark) A counter-example is enough. for example $f(n) = n^4$ and $g(n) = n^2$.(2marks)

(c) We know that k is in $O(1)$ for any constant k. Is the following claim correct? Briefly explain.

$$\sum_{k=1}^{n} k = \sum_{k=1}^{n} O(1) = O(n)$$

Solution: Incorrect. (1 mark)
$\sum_{k=1}^{n} k = n(n+1)/2$ is actually in $\Theta(n^2)$.

(d) Using the Θ notation, give the complexity of function $f(n) = 5n^3 + 4n^2$.

Solution: $\Theta(n^3)$.

(e) Using the Θ notation, give the complexity of function $f(n) = 200 \log n + \sqrt{\log n}$.

Sorting and Searching

Question 5

(a) Please illustrate the process of sorting the list $\{2, 8, 6, 1, 9\}$ using bubble sort.

[2 marks]

Solution: Bubble sort bubbles the largest number up to the end of the list. $\{2, 6, 1, 8, 9\}$ $\{2, 1, 6, 8, 9\}$ $\{1, 2, 6, 8, 9\}$ $\{1, 2, 6, 8, 9\}$ The last iteration cannot be skipped. -0.5

(b) Please illustrate the process of merging the two sorted lists $\{2, 3, 6, 8\}$ and $\{1, 2, 9, 12\}$ using mergesort.

[2 marks]

1 mark for merge w/ no clear proc.

Solution: Use two pointer to track the next integer in the two list and compare them to decide the next integer to be put into the list. (1 mark) After one list is finished, copy and paste the remaining elements in the other list to fill up the slots.(1 mark)

(c) Consider quick-sort.

i. What kind of pivot value will result in the best-case performance? What is the complexity of the best-case performance? Please give some explanation to justify your answer.

[4 marks]

Solution: When the pivot value is the median value of the list. Every iteration will split the list into two sublist of the approximately same size (2 marks). O(n log n) or Θ of that (1 mark). We will only have log n levels of spiting the list, and each level takes O(n). It is also ok if they say this complexity can be used by Master's Theorem. (1 marks)

ii. What kind of pivot value will result in the worst-case performance? What is the complexity of the worst-case performance? Please give some explanation to justify your answer.

[4 marks]

Solution: When the pivot value is either the minimal or the maximum value from the list, quicksort will simply divide the list into a sublist of size 1 and another sublist of size n-1. This corresponds to the worst case. (2marks) $O or \Theta(n^2)$ (1 mark). We will need n levels of spiting the list, and each level takes O(n). it is also ok if they explain this with discussing outer and inner loops. (1 marks)

Please go on to the next page...

(d) Describe the steps for doing a bucket sort for a list, or give a high-level pseudo-code for this.

[4 marks]

counting sort f
e.g index per
occurance

Solution: Create multiple empty buckets.
Go over the list to be sorted and scatter the objects into the buckets.
Sort each bucket separately using other sorting algorithm, e.g. insertion sort.
Visit the buckets in turn and collect the results.

(e) Write a pseudo-code for a function ***list merge(list list1, list list2)*** that takes two sorted lists, ***list1*** and ***list2***, and merges them into a new sorted list, and returns the new list. Assume that the size of list *X* can be found by *X.size* and that the given lists are sorted in ascending order.

[9 marks

Solution: Note that we request a pseudo-code. So, if the student is not using variables i and j, but the pseudo code is clear that iterates on the sublists from the first element to the last, then it should be fine.

```
make the new list listResult (1 mark)
i=0 and j=0
while (i<=list1.size and j<= list2.size) (2 marks)
{
  if(list1[i]<=list2[j]) (1 marks)
  {
    add list1[i] to listResult (1 mark)
    i++
  }
  else
  {
    add list2[j] to listResult (1 mark)
    j++
  }
}
//either list1 or list2 may still have elements
if (i<=list1.size) (1 mark)
   concatenate the rest of list1 to the result (1 mark)
if (j<=list2.size)
   concatenate the rest of list2 to the result (1 mark)
```

[Total for Question 5: 25 ma

```
    int data;
    Node *link;
}
```

(a) Please write C++ code to add one new item to the end of a sing linked list. The head of the list and the new item are given as inp parameters.

Solution: follow the linked until reaching the last node and keep it somewhere. (2 marks)
make a new node (1 mark)
set the data (1 mark)
set the link of that to null (or nullptr) (1 mark)
set the link of the last not to the address of the new node (1 mark)

(b) What does the following function return when the head of the linked list is given to it as the input? Please analyse the complexity of the function.

```
int function1(Node *head){
    if(!head)
        return;
   return  head -> data+ function1(head -> link);
}
```

Solution: returns the sum of all elements. (2 marks) complexity O(n). The function can be converted to a for loop of size n or the student may say the size of the list is reduced by One in each call of the function, and each call takes O(1).(2 marks)

(c) Stacks and Queues are often implemented based on linked lists.

i. What is a stack and what are its common operations?

Solution: Stack:LIFO
add from the top
remove from the top
check if empty

ii. Please give an application of the stack.

Solution: Checking symbol balance.
Keeping track of function calls.
Converting or Calculating postfix expressions.

[2

l) A deque is a data structure consisting of a list of items, on which the following operations are possible:

- `push(x)`: Insert item x on the front end of the deque.
- `pop()`: Remove the front item from the deque and return it.
- `inject(x)`: Insert item x on the rear end of the deque.
- `eject()`: Remove the rear item from the deque and return it.

Can you use a singly linked list to implement a deque in which the above basic operations are done in running time complexity of $O(1)$? If not, state which of these operations cannot be done in $O(1)$ time and what is the running time complexity of that/those operation(s)? Explain your design and provide some analysis for the complexity of each of all 4 operations.

Solution: Keep two pointers pointing to the front end and the rear end of the deque. (2 mark)
3 operations O(1) (1 mark each)
One operation (either eject or pop) O(n) (2 mark)

[Total for Question