1. (a) A call to the constructor (in class `Node`) is made;
This allocates storage space for the Node data;
Both data members are set to null; ***[3 marks]***

(b) *Award marks as follows up to **[3 marks max]**.*
*Award **[1 mark]** for top pointing to first node.*
*Award **[1 mark]** for names in the correct order.*
*Award **[1 mark]** for null pointer at end.*
*Award **[1 mark]** for arrows connecting each node.*

Example:

*(Award **[3 marks]** for correct diagram, arranged vertically.)* ***[3 marks]***

(c) *Award marks as follows up to **[7 marks max]**.*
*Award **[1 mark]** for specifying `pop` is a String method.*
*Award **[1 mark]** for testing for empty stack.*
*Award **[1 mark]** for calling `getName()` method correctly.*
*Award **[1 mark]** for setting `top` to next node (allow `top = top.next`).*
*Award **[1 mark]** for saving the name **before** advancing `top`.*
*Award **[1 mark]** for adjusting `top` using `getNext()` method.*
*Award **[1 mark]** for error message on empty stack.*
*Award **[1 mark]** for returning an appropriate String (i.e. correct name or empty String).*

Example:

```
public String pop()
{
  String name = "";
  if (top != null)
  {
    name = top.getName();
    top = top.getNext();
  }
  else
  {
    output("Error, nothing to pop");
  }
  return name;
}
```
[7 marks]

continued…

Question 1 continued

(d) *Award marks as follows up to **[3 marks max]**.*
*Award **[1 mark]** for loop through original array.*
*Award **[1 mark]** for correct use of `push()` (allow `push()` on its own).*
*Award **[1 mark]** for correct use of `pop()` (allow `pop()` on its own).*

Candidates may also use the condition `x < names.length` in the loops.

Example:

```
for (int x = 0; x < 6; x++)
{
  s.push(names[x]);
}
for (int x = 0; x < 6; x++)
{
  names[x] = s.pop();
}
```

[3 marks]

(e) *Award **[1 mark]** for a method and **[1 mark]** for a description up to **[2 marks]** × 2 = **[4 marks max]**.*

`isEmpty();`	returns true if the stack is empty;
`top();`	returns the top value without removing it;
`isFull();`	returns true if the stack is full;
`size();`	returns the number of items in the stack;

[4 marks]

Total: [20 marks]

1. (a) (i) *Award up to **[2 marks max]**.*
Queue is a FIFO data structure;
A list in which items may be added only at one end;
And removed only at the other end; ***[2 marks]***

(ii) *Award up to **[2 marks max]**.*
Stack is a LIFO data structure;
A list in which one of the ends is designated as the top of the stack;
And access (store and retrieve) is restricted to this end of the list; ***[2 marks]***

(b) *Example answers:*
Transfer of data from/to I/O devices;
Simulation;
Job queue, order of processing; ***[1 mark]***

(c) 5; ***[1 mark]***

(d) (i) The queue is empty! ; ***[1 mark]***

(ii) *Award marks as follows up to **[4 marks max]**.*
*Award **[1 mark]** if the queue is not empty;*
*Award **[1 mark]** for temporarily storing the value of the item;*
*Award **[1 mark]** if this is at the beginning of the queue;*
*Award **[1 mark]** for changing the pointer that points to the end of the queue if the item to be deleted is the last one;*
*Award **[1 mark]** for changing the value of the pointer that points to the beginning of the queue;*
*Award **[1 mark]** if it points to the next item in the queue;*
*Award **[1 mark]** for returning the value that was at the beginning of the queue;*

```
public int dequeue()
{
  if (isEmpty())
  {
    output ("Queue empty");
    return -1;
  }
  else
  {
    int temp=first.item;
    if (first.next== null) // only 1 item in queue
    { last = null;}
    first = first.next // first node removed
    return temp:
  }
}
```

[4 marks]

continued ...

Question 1 continued

(iii) *Award marks as follows up to **[3 marks max]**.*
*Award **[1 mark]** for deleted item 2.*
*Award **[1 mark]** for three items in the queue.*
*Award **[1 mark]** for all three correct values (4, 1, 7).*

Deleted item: 2
Items in the queue: 4, 1, 7 ***[3 marks]***

(e) *Award up to **[6 marks max]**.*
Initialize an empty stack;
While queue is not empty;
Remove the element from the beginning of the queue/dequeue;
Push the removed element onto the stack;
While stack is not empty;
Pop an element off the stack;
Display it/enqueue it;

Possible answer:
Take the items off the queue;
And put them one by one;
In a new;
Stack;
Now take them off the stack;
And put them back in the queue; ***[6 marks]***

Total: [20 marks]