May 2013 – Computing Advanced

Paper 1

Section A

Question 1

The output is the following:

SuperClass: Constructor
SubClass: Constructor
SubClass: methodA 4
SuperClass: methodX

Question 2

The code will not execute because one cannot assign a superclass object to a subclass object as is done in the third line of code. For the code to be correct one should substitute SubClass in line three with SuperClass.

Section B

Question 3

Address	Data
401	54
402	42
403	43
404	56

R1 | 0 |

R2 | 0 |

MOV R1, 02

Address	Data
401	54
402	42
403	43
404	56

R1 | 02 |

R2 | 0 |

MOV R2, 401

Address	Data
401	54
402	42
403	43
404	56

R1 | 02 |

R2 | 401 |

ADD R2, R1

Address	Data
401	54
402	42
403	43
404	56

R1	02
R2	403

MOV (R2), RI

Address	Data
401	54
402	42
403	02
404	56

R1	02
R2	403

Question 4

	Best case	Worst case
Bubble Sort	O(n)	$O(n^2)$
Quick Sort	O(n log n)	$O(n^2)$

As can be seen from the table the bubble sort is faster in the best case. In the worst case both algorithms are slow.

Question 6

(a) Sequential file, random file, indexed sequential file.
(b) In a sequential file the records are organized in such a way that they can only be read sequentially from start to finish. Some programs, for example payroll, work faster with such files.

Question 9

(a) 011111.11
(b) 010100.00

Paper 2

Question 4

(a)

A process can be in one of three states:

- Executing
- Ready (waiting)
- Blocked (suspended)

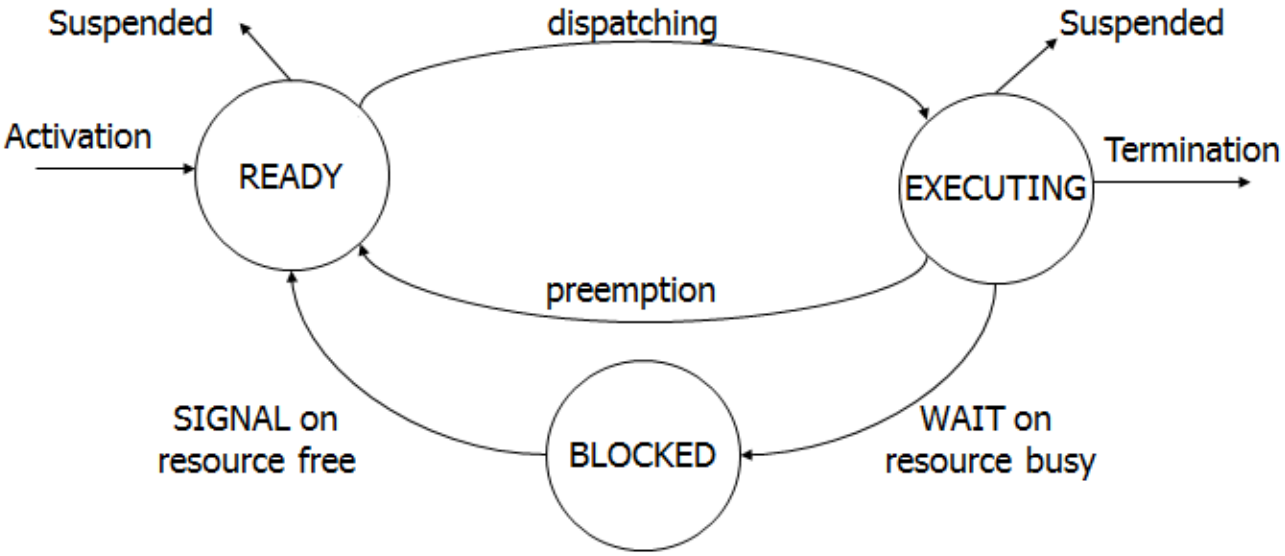

Diagram 1. The three states of a process.

Moving from one State to Another	**Executing**	**Ready**	**Blocked**
Executing	-----	Pre-emption stops temporarily a process.	The process stops because it is waiting for an event to happen e.g. input.
Ready	The dispatcher sends an order for the execution of the process.	-----	-----
Blocked	-----	The event that the process was waiting for to continue has happened.	-----

(b)

A deadlock occurs when two or more processes are blocking each other. As an example, suppose that process A is holding resource X and it is waiting for the release of resource Y and at the same time process B is holding resource Y and is waiting for the release of resource X to continue. This causes a deadlock.

(c) How a deadlock can be minimized

Some ways which can be employed to reduce the chances of deadlock occurrence are the following:

- Processes should try to access resources in the same order. If this were to be done deadlocks will be avoided (however there can be circumstances when this measure cannot be assured).
- Avoid as much as possible user interaction during critical regions. This measure will reduce the chances of a deadlock.
- Keep critical regions as short as possible. The shorter they are the much likely it is that deadlocks will not occur.
- A process should hold (lock) simultaneously the least number of resources possible.

Question 7

(a)

```
push (ST, x) //ST is the stack, x is the element to be pushed
begin
    read T //T is the pointer to the top of the stack
    if (there is more space for new elements)
        then
            begin
                T = T + 1
                place x at T
                return T
            end
        else
            return -1 //stack is full
end

pop (ST)
begin
    read T
    if (T = -1) //stack is empty
        then
            return null
        else
            begin
                E = element at T
                T = T – 1
                return E
            end
end
```

(b)

```
layoutContainers (int n)
/* LL is a linked list such that each element holds a stack of not more than five
elements */
```

```
begin
if (n>0) then
        begin
            pile = last element of LL;
            while (pile is not full) and (n>0) do
                begin
                    c_num = container number
                    push c_num in pile
                    n - -
                end
        end
    while (n>0) do
        begin
            pile = create new stack at end of linked list
            while (pile is not full) and (n>0) do
                begin
                    c_num = container number
                    push c_num in pile
                    n - -
                end
        end
end
```

(c)

```
loadShips (n)
begin
    while (n>0) do
        begin
            pile = last element of LL
            while (n>0) and (pile is not empty) do
                begin
                    c_num = pop from pile
                    transfer container c_num to ship
                    n - -
                end
            if (pile is empty)
                then eliminate pile from LL
        end
end
```

(d)

```
searchContainerLocation (num)
```

```
begin
    stack = 0
    position = 0
    found = “no”
    while (found = “no”) do
        begin
            if (there is another pile to search)
                then
                    begin
                        pile = next element of LL
                        stack ++
                        position = 0
                        st_temp = new stack
                        stack_empty = “no”
                        while (found = “no”) and (stack_empty = “no”) do
                            begin
                                    cont_num = pop (pile)
                                    push (st_temp, cont_num)
                                    position++
                                    if (cont_num = num) then found = “yes”
                            end
                        pop elements from st_temp and push them in pile
                    end
        end
    if (found = “yes”)
        then display (stack, position)
        else display (“not found”)
end
```
