(ANSWER ANY FOUR QUESTIONS)

Question 1

a) Is the given code fragment correct?

```
|[
    var x: int;
     ...
   {x < 3}
   x := x + 2;
   x := x - 7
   {x < -3}
]|
```

(5 marks)

b) Write down an invariant **P** for the given loop and prove that the body of the loop is correct. Your answer must also show program termination.

```
|[ con N : int;    {N ≥ 0}
   var  x : int;
   x : = 0;
    do (x + 1) * (x + 1) ≤ N →
        {P ∧ (x + 1)² ≤ N}
         x : = x + 1
        {P}
    od
   { x² ≤ N < (x + 1)² }
]|
```

(10 marks)

c) Using an invariant diagram derive an $O(N)$ solution to the following problem. Your answer should include a complete solution.

```
|[ con N : int; { N ≥ 0}
  var f : array[0..N) of bool;
      n : int;
  S
  {0 ≤ n ≤ N ∧ ( ∀j : 0 ≤ j < n : f.j = true)
              ∧ ( ∀j : n ≤ j < N : f.j = false)}
]|
```

Note : Only swap operations are allowed on f.

(10 marks)
[Total 25 marks]

Question 2

Write down the invariants **P0** and **P1** which describe the program below and hence derive the programs formal proof. An annotated program should be included in your answer along with a proof for program termination.

```
|[ con N : int; { N ≥ 0}
       f : array[0..N) of boolean;
   var n : int;
       b : bool;
   b,n := true,0;
   do n < N →
       b := b ∧ f.n;
       n := n + 1;
   od
   { b ≡ (∀i : 0 ≤ i < N : f.i) }
]|
```

[25 marks]

Question 3

Formally derive a solution to the given specification. Your answer should include a complete solution.

```
|[ con
     N : int; { N ≥ 0 }
    f : array[0 .. N ) of int;
   var
    x, freq : int;
   S
   {  x = maxj : 0 ≤ j < N : f.j ∧ freq = #j: 0 ≤ j < N : f.j = x}
]|
```

[25 marks]

Question 4

Write a specification and derive a solution for the following problem. You answer must include a complete solution.

> Given a character array f [0..N), N ≥ 0, determine if the array contains at least one '*'.

[25 marks]

Question 5

Using the specification below, formally derive the sorting algorithm known as selection sort.

```
|[ con N: int; { N ≥ 0 }
   var
     f : array[0 .. N ) of int;
     Sort
     { ∀i : 0 ≤ i < N: (∀j : i ≤ j < N : f.i ≤ f.j) }
]|
```

Note: You are only allowed to swap elements in f, thereby ensuring that the final array is a permutation of the original.

[25 marks]

Laws of the Calculus

Let P, Q, R be propositions

1. Constants

   $P \vee true \equiv true$

   $P \vee false \equiv P$

   $P \wedge true \equiv P$

   $P \wedge false \equiv false$

   $true \Rightarrow P \equiv P$

   $false \Rightarrow P \equiv true$

   $P \Rightarrow ture \equiv true$

   $P \Rightarrow false \equiv \neg P$

2. Law of excluded middle : $P \vee \neg P \equiv true$
3. Law of contradiction: $P \wedge \neg P \equiv false$

4 Negation : $\neg\neg P \equiv P$

5. Associativity: $P \vee (Q \vee R) \equiv (P \vee Q) \vee R$

   $P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$

6. Commutativity: $P \vee Q \equiv Q \vee P$

   $P \wedge Q \equiv Q \wedge P$

7. Idempotency: $P \vee P \equiv P$

   $P \wedge P \equiv P$

8. De Morgan's laws : $\neg (P \wedge Q) \equiv \neg P \vee \neg Q$

   $\neg (P \vee Q) \equiv \neg P \wedge \neg Q$

9. Implication $P \Rightarrow Q \equiv \neg P \vee Q$

   $P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P$

   $(P \wedge Q) \Rightarrow R \equiv P \Rightarrow (Q \Rightarrow R)$

10. (If and only if) $\equiv$: $P \equiv Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$
11. Laws of distribution: $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

    $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

12. Absorption: $[P \wedge (P \vee R) \equiv P]$

    $[P \vee (P \wedge R) \equiv P]$

13. **Predicate Calculus**

Negation

$\forall x \neg P(x) \equiv \neg \exists x P(x)$

$\exists x \neg P(x) \equiv \neg \forall\ P(x)$

$\exists x P(x) \equiv \neg(\forall x \neg P(x))$

Universal Quantification

$[(\forall x : P(x)) \wedge (\forall x : Q(x)) \equiv (\forall x : P(x) \wedge Q(x))\ ]$

$[(\forall x: P(x)) \vee (\forall x : Q(x)) \Rightarrow (\forall x: P(x) \vee Q(x))\ ]$

$[Q \vee (\forall x : P(x)) \equiv (\forall x: Q \vee P(x))\ ]$, where x not free in Q

$[Q \wedge (\forall x: P(x)) \equiv (\forall x: Q \wedge P(x))\ ]$, where x not free in Q

Existential Quantification

$[(\exists x: P(x) \wedge Q(x)) \Rightarrow (\exists x : P(x)) \wedge (\exists x : Q(x))\ ]$

$[(\exists x : P(x)) \vee (\exists x: Q(x)) \equiv (\exists x: P(x) \vee Q(x))\ ]$

$[Q \vee (\exists x: P(x)) \equiv (\exists x : Q \vee P(x))\ ]$, where x not free in Q

$[Q \wedge (\exists x: P(x)) \equiv (\exists x : Q \wedge P(x))\ ]$, where x not free in Q

$[\ (\exists x : P(x)) \equiv \neg(\forall x: \neg P(x))]$

$[\ (\neg\exists x : P(x)) \equiv (\forall x: \neg P(x))]$

14. **Universal Quantification over Ranges**

$[\forall i : R : P \equiv \forall i : \neg R \vee P\ ]$ Trading

$[\forall i : false : P \equiv true\ ]$

$[\forall i : i = x : P \equiv P(i := x)\ ]$ One-point rule

$[(\forall i : R : P) \wedge (\forall i : R : Q) \equiv (\forall i : R : P \wedge Q)\ ]$

$[(\forall i : R : P) \wedge (\forall i : S : P) \equiv (\forall i : R \vee S : P)\ ]$

$[(\forall i : R : P) \vee (\forall i : R : Q) \Rightarrow (\forall i : R : P \vee Q)\ ]$

$[Q \vee (\forall i : R : P) \equiv (\forall i : R : Q \vee P)\ ]$

$[Q \wedge (\forall i : R : P) \equiv (\forall i : R : Q \wedge P)\ ]$

15. **Existential Quantification over Ranges**

$[\ \exists i : R : P \equiv \exists i : R \wedge P\ ]$ Trading

$[\ \exists i : false : P \equiv false]$

$[\ \exists i : i = x : P \ \equiv\ P(i := x)\ ]$ One-point rule

$[(\exists i : R : P \wedge Q) \Rightarrow (\exists i : R : P) \wedge (\exists i : R : Q)\ ]$

$[(\exists i : R : P) \vee (\exists i : R : Q) \equiv (\exists i : R : P \vee Q)\ ]$

$[Q \vee (\exists i : R : P) \equiv (\exists i : R : Q \vee P)\ ]$

$[Q \wedge (\exists i : R : P) \equiv (\exists i : R : Q \wedge P)\ ]$