Question 1

```
function func(param) {
    for (set i = 0 ,i < assignment.length(),i++) do
        char key_word = assignment[i];// key_word meaning the each one element in the mathmatical expression
        if (key_word = "(" or "[" or "{") {
            stack.puch(key_word);
        } //if key_word is "(" or "[" or "{" push to the stack
        if (key_word = ")" or "]" or "}")
        char recheck = stack.pop() // recheck meaning the which one of the element first pop in the stack
            if (recheck == "(" and key_word == ")") {
                stack.pop(); // if matching pop the first element in the stack
            }
            if (recheck == "[" and key_word == ")") {
                stack.pop();
            }
            if (recheck == "{" and key_word == ")") {
                stack.pop();
            }
            else { //if the element can not matching first one in the stack that meaning have a mistake like (] that
                print "miss a" recheck "in this mathematical expression on the location " i // recheck meaning wich kind of parentheses and
                return False                                                                //i meaning the location in the mathmatical
            }

        }
    end
// if all have matching that meaing the stack should be empty
if (stack.isempty()) {
    return True
}
// if the stack still have one char that meaning this mathmatical expression have one more useless "(" or "[" or "{" at the begin
else{
    char miss_one = stack.pop;
    print "you hava one more " miss_one "can not find the matching in this mathematical expression"
    return False
}
}
```

The most of time will be on the fist one for because he need check each one of the math expression and during the for also have a lot of if that if also the time spend point.

Question2

(a) $(65n^4 + 2n + 3)/(n + 1) = \Theta(n^3)$ [2 marks]

The largest element is 65n^4/(n+1) in this math expression that meaning n' s degree is 3 and we need ignore the coefficients that meaning that will be equal to n^3 and that meaning its is equal to the theta(n^3), so that is true

(b) 21nlogn+2n+1=Θ(nlogn)[2marks]

That is same with (a) the 21nlogn is the largest element in this math expression and we also need ignore the coefficient too that meaning we will get the answer is equal to theta(nlogn), so that is true

Question3

(a) $n^2 = \Theta(\log n)$

By the step of question the answer od n^2 should be theta(n^2) and on the other hand n^2 and log(n) not the relationship of f(n) and theta(n)

(b) $n^n = \Theta(2^n)$ [3 marks]

For this question I think he have a special n that is 2 but when n > 2 f(n) != theta (n)

Question 4

```
1
2   Selectionsort(double_linked_list list) {
3       if (list.head == null) {//if the list is null that meaning we dont need do anything.
4           return list;
5       }
6       node list_long = list.head;
7       int long = 1; //because we need use next to get the length of the list of we can not begin from 0 should begin from 1
8       while (list_long.next != null) {
9           long++;
10          list_long = list_long.next
11      } // get the long of the list thus know the loop time
12      for (int i = 0, i < long, i ++) {
13          node front =list.head
14          node list_next =list.next
15          while (list_next != null) {
16              if (list_next.value<front.value){      //  front = 5 . 4 . 6 list_next= 4 . 6
17                  front.next =list_next.next         //  front = 5 . 6     list_next= 4 . 6
18                  list_next.next =front.head         //  front = 5 . 6     list_next= 4 . 5 . 6
19                  node buffer = front . head         //  front = 5 . 6     list_next= 4 . 5 . 6 buffer= 5 . 6
20                  front .head = list_next .head      //  front = 4 . 5 . 6 list_next= 4 . 5 . 6 buffer= 5 . 6
21                  list_next.head =buffer .head       //  front = 4 . 5 . 6 list_next= 5 . 6     buffer= 5 . 6
22                  buffer.clear                       //  front = 4 . 5 . 6 list_next= 5 . 6
23                  list_next =list_next.next          //  front = 4 . 5 . 6 list_next= 6
24                  front =front.next                  //  front = 5 . 6     list_next= 5 . 6
25              }
26              // one time check finally
27          }
28          // why we need this loop for the how much element time
29          //because maybe a minimum on the right least one and that only one time each one time
30          //loop so that at least need long-1 time the least one time just for check
31
32      }
33      // when the everyone element we will get a new one sort list
34      return list;
35  }
```