3.1

CCDAA DBBCD DDBCB

3.2.4

```
#include <iostream>
#include <stack>

using namespace std;

int evalRNP(char *token)
{
    stack<int> stack;
    for (int i = 0; token[i] != '$'; i++)
    {
        if (isdigit(token[i]))
        {
            int num = int(token[i]);
            while (isdigit(token[i + 1]))
            {
                num = num * 10 + int(token[i]);
                i++;
            }
            stack.push(num);
        }
        else if (token[i] == '+')
        {
            int a = stack.top();
            stack.pop();
            int b = stack.top();
            stack.pop();
            stack.push(a + b);
        }
        else if (token[i] == '-')
        {
            int a = stack.top();
            stack.pop();
            int b = stack.top();
            stack.pop();
            stack.push(a - b);
        }
        else if (token[i] == '*')
        {
            int a = stack.top();
            stack.pop();
            int b = stack.top();
            stack.pop();
            stack.push(a * b);
        }
        else if (token[i] == '/')
        {
            int a = stack.top();
            stack.pop();
            int b = stack.top();
            stack.pop();
            stack.push(a / b);
        }
    }
    int res = stack.top();
    return res;
}
```

3.2.4

(1)

```
#include <iostream>

int ackerman(int m, int n)
{
    if (m == 0)
        return n + 1;
    else if (n == 0)
        return ackerman(m - 1, 1);
    else
        return ackerman(m - 1, ackerman(m, n - 1));
}
```

(2)

由 f(1, 1) = 1， f(2, 1) = 2， f(1, 2) = f(2, 2) = 3。 可知， f(1, 2) = f(1, f(1, 1)) + f(1, 1) = 1 + f(1, 1) = 2。 f(1, f(1, 1)) = f(1, 1) = 1。 因此， f(1, 2) = 2。

(3)

```
int ackerman(int m,int n){
    int a[M][N];
    int i,j;
    for(j=0;j<N;j++){
        a[0][j]=j+1;
    }
    for(i=1;i<m;i++){
        for(j=1;j<N;j++){
            a[i][j]=a[i-1][a[i][j-1]];
        }
    }
    return(a[m][n]);
}
```

3.2.10

(1)

```
int findMax(Node* head, int max) {
    if (head == NULL) {
        return max;
    }
    if (head->data > max) {
        max = head->data;
    }
    return findMax(head->next, max);
}
```

(2)

```
int length(LinkList L)
{
    if(L->next==NULL){
        return 0;
    }else{
        return length(L->next)+1;
    }
}
```

(3)

```
double findAvg(Node* head, double sum, int count) {
    if (head == NULL) {
        return sum / count;
    }
    sum += head->data;
    count++;
    return findAvg(head->next, sum, count);
}

findAvg(head, 0, 0)即为平均值
```