进制转换

```cpp
#include <iostream>
#include <stack>
using namespace std;

int main(){
    stack<int> s;
    int n;
    scanf("%d",&n);
    while(n!=0){
        s.push(n%2);
        n=n/2;
    }
    while(s.empty()!=1){
        cout<<s.top();
        s.pop();
    }

}
```

1353：表达式括号匹配(stack)

```cpp
#include <iostream>
#include <string>
#include <stack>
//括号匹配
using namespace std;
stack <int> s;
int main(){
    string v;
    bool flag=false;
    cin>>v;
    for (int i=0;i<=v.length();i++){
        char t=v[i];
        if (t=='@')
            break;
        if (t=='(')
            s.push(1);
        else if (t==')'&&s.size())
            s.pop();
        else if (t==')'&&!s.size())
            flag=true;
    }
    if (s.size())
        flag=true;
//    cout<<flag;
    string ans=flag?"NO":"YES";
    cout<<ans<<endl;
    return 0;
}
```

1354：括弧匹配检验

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<stack>
//括号匹配
using namespace std;
char str[10000];
stack<int> S;//用1代表(，2代表)，3代表[，4代表]
int main(){
    cin>>str;
    int len=strlen(str);
    for(int i=0;i<len;i++)
    {
        if(str[i]=='(')//记录左圆括号
            S.push(1);
        else if(str[i]==')')//记录右圆括号
        {
            if(S.empty())//栈为空
                S.push(2);
            else if(S.top()==1)
                S.pop();
            else
                S.push(2);
        }
        else if(str[i]=='[')//记录左方括号
            S.push(3);
        else if(str[i]==']')//记录右方括号
        {
            if(S.empty())//栈为空
                S.push(4);
            else if(S.top()==3)
                S.pop();
            else
                S.push(4);
        }
    }
    if(S.empty())
        printf("OK");
    else
        printf("Wrong");
    return 0;
}
```

1331：【例1-2】后缀表达式的值

```cpp
//后缀表达式求值
#include <iostream>
#include <cstring>
using namespace std;
int stack[101];
char s[256];
int compute(char s[256]){
```

```
    int i=0,top=0,x;
    while(i<=strlen(s)-2){
        switch (s[i]){
            case '+':stack[--top]+=stack[top+1];break;//int
temp=stack[top+1]+stack[top];top--;stack[top]=temp;
            case '-':stack[--top]-=stack[top+1];break;
            case '*':stack[--top]*=stack[top+1];break;
            case '/':stack[--top]/=stack[top+1];break;
            default:
                x=0;
                while(s[i]!=' '){
                    x=x*10+s[i++]-'0';
                }
                stack[++top]=x;
                break;
        }
        i++;
    }
    return stack[top];
}
int main(){
    cin.getline(s,256);
    cout<<compute(s);
    return 0;
}

```

1369：合并果子(fruit)

```
#include <iostream>
#include <queue>
#include <cstdio>
using namespace std;
int n;
priority_queue<int,vector<int>,greater<int> > h;//升序排列，小顶堆
//priority_queue<int, vector<int>, less<int> > a;降序排列，大顶堆
void work(){
    int i,x,y,ans=0;
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>x;
        h.push(x);
    }
    for(i=1;i<n;i++){
        x=h.top();h.pop();
        y=h.top();h.pop();
        ans+=x+y;
        h.push(x+y);
    }
    cout<<ans<<endl;
}

int main(){
    work();
```

```
    return 0;
}

```

单调队列

队列：先进先出

优先队列：优先级高的元素进队时插队

单调队列：队列始终满足从队首开始，优先级递减，进队时间越靠后越晚；新元素进队时，将优先级低于自己的直接踢出队。

应用：动态规划优化

1597：【例 1】滑动窗口

核心思想：高效地排除不可能成为答案的数字。

后来的数字大于前面的数字，则前面的数字不可能再成为答案。

每个数字进队时踢出前面比自己大的数，始终取队首为答案。

单开一个数组记录队里数在原数组的下标，队首数字太老就出队。

P1886 滑动窗口

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxx=1000001;
long long int N,K,data[maxx],dl[maxx],id[maxx],le=1,ri;
int main(){
    scanf("%lld%lld",&N,&K);
    for(long long int i=1;i<=N;i++)
        scanf("%lld",&data[i]);
    for(long long int i=1;i<=N;i++){
        while(le<=ri&&data[i]<dl[ri])
            ri--;
        ri++;
        dl[ri]=data[i];
        id[ri]=i;
        if(id[le]+K<=i)
            le++;
        if(i>=K)
            printf("%lld ",dl[le]);
    }
    printf("\n");
    memset(dl,0x3f3f3f3f,sizeof(dl));
    le=1,ri=0;
    for(long long int i=1;i<=N;i++){
        while(le<=ri&&data[i]>dl[ri])
            ri--;
        ri++;
        dl[ri]=data[i];
        id[ri]=i;
        if(id[le]+K<=i)
            le++;
```

```
        if(i>=K)
            printf("%lld ",dl[le]);
    }
    printf("\n");
}
```