P1

Function AmountofCoins(n)//n is the total amount of money in cents

Create an array A[5], each element is 0//each element in A[5] represents the number of the specific coins.

Create an array B[5]={100,25,10,5,1}

i=1

While(n>0)

A[i]=int(a/B[i])

n=n%B[i]

i=i+1

P2

Here are the c++ code:

```
#include <iostream>

using namespace std;

int main()
{
	int i;
	int j;
	int n;
	cout<<"please input a number n"<<endl;
	cin>>n;
	int C[n]={0};
	int open=0;
	int closed=0;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
			if(j%i==0)
			C[j]=C[j]+1;
		}
	}
	cout<<endl;
	for(i=1;i<=n;i++)
	{
		cout<<C[i]<<endl;
	}
	cout<<endl;
	for(i=1;i<=n;i++)
	{
```

```
            if(C[i]%2==0)
                closed=closed+1;
            else
                open=open+1;
        }
        cout<<open<<endl;
        cout<<closed<<endl;
        return 0;
}
```

(a)
After coding, it is clear to find that if the final status of a door is closed, it should be toggled even times, and on the other hand if the final status of a door is open, it should be toggled odd times. From 1 to n, only if a is the square of another integer, a can have odd number of factors.i.e(a=b^2)

Thus only the doors whose number of factors are odd are left open finally, and the other doors are open.

(b)
According to (a), we can find the doors open with number k

K=1,4,...$\left\lfloor \sqrt[2]{n} \right\rfloor^2$

3
(a)

```
#include <iostream>
using namespace std;
int maxsimum(int a[], int l, int r) {
        if (l == r)
            return a[l];
        int m = (l+r)/2;
        int u = maxsimum(a,l,m);
        int v = maxsimum(a,m+1,r);
        return u>v?u:v;
}

int main() {
        int a[] = {34,23,45,99,30,31,57,33,55,10};
        int n = sizeof(a)/sizeof(int);
        cout << maxsimum(a,0,n) << endl;
        return 0;
}
```

(b)
If n=1
T(n)=Θ(1)
T(n)=2T(n/2)+ Θ(1)
T(n)∈ $\Theta(n)$

(c)
Initialization p=r
A[p] is the only element in array
A[p] is the maximum value

Maintenance
2 recursive call will return 2 maximum value separately
1 of the 2 maximum value will be returned

Termination
The maximum value of the n elements will be returned after comparing the two separate maximum values from sub-problems.

(d)
To find the maximum value in n numbers, it is necessary to look at all numbers in the array.
Thus the lower bound is $\Omega(n)$
The upper bound is O(n)
Thus this function is optimal