Problem1:

1. False
2. False
3. True
4. False
5. True

Problem2:

1(a). int j = 5;  
while(j < log n){

int k = 5; while(k < n){

sum += a[j] \* b[k];

k = k1.3; }

j = 1.3 \* j; }

Time Complexity: loglog(n)\*loglog(n)

1(b). for(int i = 1 to n){ for(int j = i to n){

for(int k = j to n){  
sum += a[i] \* b[j] \* c[k];

}  
if(gcd(i, j) == 1){

j = n; }

} }

Time Complexity: n\*n\*n

2(a). j = 1; while(j < n){

k = j; while(k < n){

sum += a[k] \* b[k];

k += log log n; }

j = j \* j; }

Time Complexity: logn\*n/(loglogn)

2(b). int j = 2; while(j < n){

int k = j; while(k < n){

sum += a[k] \* b[k];

k += n1/3 logn; }

j = j \* √5; }

Time Complexity: n^(2/3)

3(a). for(int i = 1 to n){ for(int j = 1 to log i){

sum += a[i] \* b[j]; }

}

Time Complexity: n\*logn

3(b). for(int i = 1 to n){  
for(int j = 1; j < n; j += i){

sum += a[i] \* b[j]; }

}

Time Complexity: n\*n

4(a). for(int i = n; i > 0; i /= 2){ for(int j = 0; j < i; j++){

sum += a[i] \* b[j]; }

}

Time Complexity: logn \* n

4(b). for(int i = 1 to n){ for(int j = 1 to n){

if((j % 2) == 0){ continue;

} else{

sum += a[i] \* b[j]; }

} }

Time Complexity: n\*n

5(a). int max(A, n){  
if(n == 1) return A[0];  
return larger of A[n-1] and max(A, n-1)

}

Time Complexity: n

5(b). function T(int n){ int n1 = T(n/2);

int n2 = n12;  
int sum = 0; for(int i = 1 to n){

for(int j = 1 to n){  
sum += i\*n1 + j\*n2;

} }

}

Time Complexity: Infinity

6(a). long power(long x, long n){ if(n == 0) return 1;  
if(n == 1) return x;  
if((n % 2) == 0)

return power(x\*x, n/2); else

return power(x\*x, n/2) \* x; }

Time Complexity: logn

fib(a){  
if((a == 1) || (a == 0)) return 1; return fib(a-1) + fib(a-2);

}

Time Complexity: 2^n

Problem3:

Queue q

int tmp=dequeue();

int min=tmp;

q.enqueue(tmp);

for (int i=0; i<queue.size();i++){

tmp=dequeue();

if(min>tmp){

min=tmp;}

queue.enqueue(tmp);}

for(int i=0; i<queue.size();i++){

tmp=dequeue();

if(tmp!=min){

queue.enqueue(tmp);}

else{ break;}}

Problem4:

**bool** return\_value=**false**;

if(s[0]== “(” &&s[s.length-1]== “)” || s[0]== “[” && s[s.length-1]== “]”|| s[0]== “{” && s[s.length-1]== “}”){

return\_value=true;}

**return** return\_value;

Problem5:

Level\_node\_total(node \*n, int level, int k){

if(n==NULL){return 0;}

if(level==k){return 1;}

return level\_node\_total(n->left, level+1,k)+ node\_at\_level(n->right, level+1,k);}

BTree \*tree

Node \*root=tree->root

int arr[N];

for (int i=0; i<N; i++){

arr[i]=level\_node\_total(root, 0, i);}