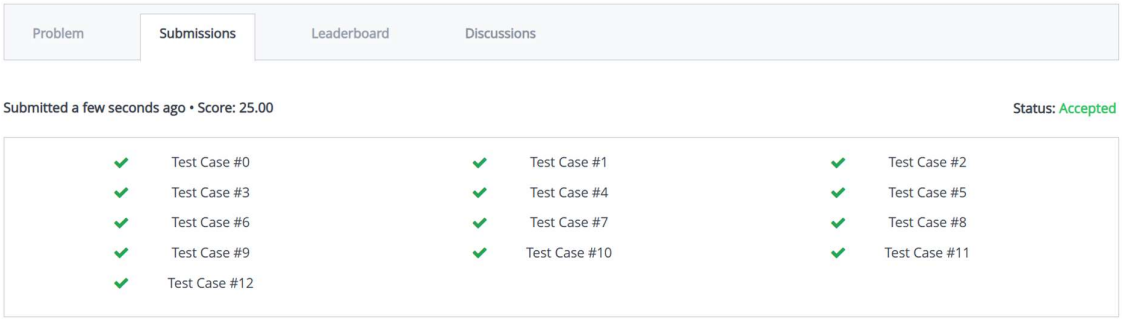
ANIL AMBATI

2300031781

DAA SKILL WEEK-13

1)

Red John is Back



SUBMITTED CODE:

```
IMPORT JAVA.IO.*;

IMPORT JAVA.UTIL.*;


PUBLIC CLASS SOLUTION {


    PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

        SCANNER SCANNER = NEW
SCANNER(SYSTEM.IN);
```

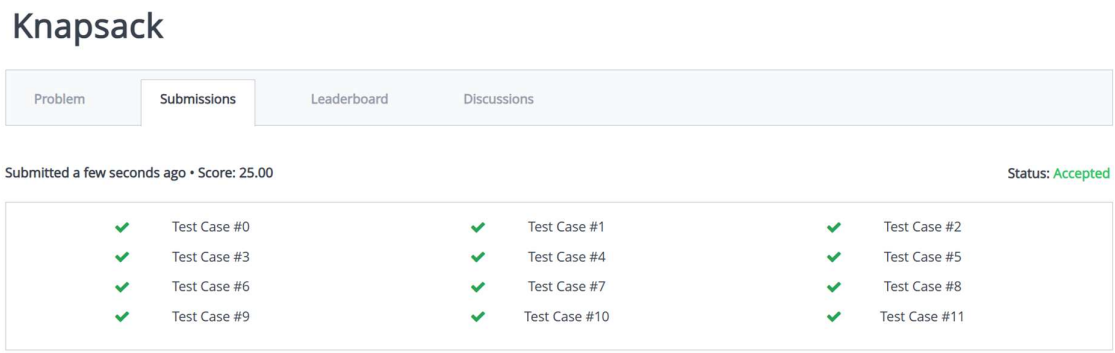
```
        // Number of test cases
        int testCases = scanner.nextInt();

        // Array to hold computed values for different positions up to 40
        int computedValues[] = new int[41];
        computedValues[1] = 1;
        computedValues[2] = 1;
        computedValues[3] = 1;
        computedValues[4] = 2;

        // Calculate number of ways to represent n using the defined recurrence
        for (int i = 5; i <= 40; i++) {
            computedValues[i] = computedValues[i - 4] + computedValues[i - 1];
        }
```

```
        // Maximum value calculated
        int maxComputedValue =
computedValues[40];

        // Array to identify prime numbers up
to maxComputedValue
        boolean isPrime[] = new
boolean[maxComputedValue + 1];

        // Initialize the isPrime array
        for (int i = 2; i <= maxComputedValue;
i++) {
            isPrime[i] = true;
        }

        // Sieve of Eratosthenes to find all
primes up to maxComputedValue
        for (int i = 2; i <=
Math.sqrt(maxComputedValue + 1); i++) {
            if (isPrime[i]) {
```

```
            for (int j = i * i; j <=
maxComputedValue; j += i) {
                isPrime[j] = false;
            }
        }
    }

    // array to count the number of
primes up to each index
    int primeCount[] = new
int[maxComputedValue + 1];

    // count cumulative number of primes
    for (int i = 2; i <= maxComputedValue;
i++) {
        if (isPrime[i]) {
            primeCount[i] = 1;
        }
        primeCount[i] += primeCount[i - 1];
    }
```

```
        // process each test case
        while (testCases-- > 0) {
            int n = scanner.nextInt();
            // output the count of primes for the computed value of the given n

System.out.println(primeCount[computedValues[n]]);
        }

        scanner.close(); // close the scanner
    }
}
```

2)



SUBMITTED CODE:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

struct qnode
    {
    int data;
    struct qnode*next;

};

struct queue
    {
   struct qnode*front,*rear;

};
```

```
struct qnode*newnode(int k)
   {
   struct qnode*temp=(struct
qnode*)malloc(sizeof(struct qnode));
   temp->data=k;
   temp->next=null;
   return temp;

};

struct queue*createque()
   {
   struct queue*q=(struct
queue*)malloc(sizeof(struct queue));
   q->front=q->rear=null;
   return q;
};

void enqueue(struct queue*q,int k)
```

```
    {
    STRUCT QNODE*TEMP=NEWNODE(K);
    IF(Q->FRONT==NULL)
       {
       Q->FRONT =Q->REAR=TEMP;
    }
    ELSE
       {
       Q->REAR->NEXT=TEMP;
       Q->REAR=TEMP;

    }
}

INT DEQUEUE(STRUCT QUEUE*Q)
    {
    IF(Q->FRONT==NULL)
       RETURN -1;
    INT TEMP=Q->FRONT->DATA;
```

```
    Q->FRONT=Q->FRONT->NEXT;
    IF(Q->FRONT==NULL)
        Q->REAR=NULL;
    RETURN TEMP;

}




INT MAIN() {

    INT T;
    INT N,K;
    INT RES,I,TEMP,TEMP2;
    INT ARR2[2003];INT J,NUM;
    INT *ARR;
    INT FOUND,FOUND1;
```

```
SCANF("%D",&T);
WHILE(T--)
   {
   FOR(I=0;I<2003;++I)
      ARR2[I]=0;
   FOUND=0;
   SCANF("%D%D",&N,&K);
   TEMP=RES=K;
   ARR=(INT*)MALLOC(N*SIZEOF(INT));
   J=0;
   FOUND1=0;
   FOR(I=0;I<N;++I)
   {



      SCANF("%D",&NUM);
      IF(K%NUM==0)
       {   FOUND1=1;
        FOUND=1;}
```

```
        IF(ARR2[NUM]==0)
          {
          ARR2[NUM]=1;
          ARR[J]=NUM;
          J++;
        }
      }


      STRUCT QUEUE*Q=CREATEQUE();
      ENQUEUE(Q,K);
      WHILE(((TEMP2=DEQUEUE(Q))!=-1) && FOUND==0)
          {
         // PRINTF("TEMP2=%D",TEMP2);
          FOR(I=0;I<J;++I)
          {
          TEMP=TEMP2-ARR[I];
```

```
        IF(TEMP<0)
            CONTINUE;
        IF(RES>TEMP)
            RES=TEMP;

        IF(RES==0)
        {
            FOUND=1;
            BREAK;
        }
        ELSE
            ENQUEUE(Q,TEMP);

    }
}

IF(FOUND&&FOUND1)
    PRINTF("%D\N",K);
```

```
        ELSE IF(RES==K)
            PRINTF("0\N");
        ELSE
        PRINTF("%D\N",K-RES);



    }


    RETURN 0;
}
```