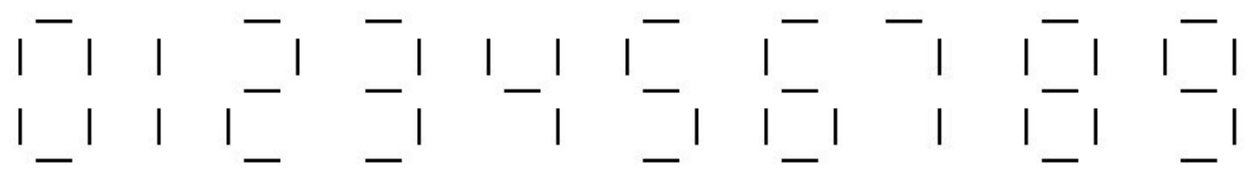
Namma RCB

Problem : Gopal has become the official scorer of RCB. He has a 7-segment display like this.



But the problem is only **N** diodes work. Now he can place the diodes wherever he wants. Gopal is a very huge fan of RCB. So he wants to display huge runs for RCB. Gopal wants to know what is the maximum number that he can display using atmost **N** diodes without repition of the digits. If no number could be displayed then print -1.

Difficulty : Medium

Pre-Requisites : Dynamic Programming

Explanation :

Approach 1: Since the digits cannot be repeated there can be 2^10 different numbers to check. Generating all possible subsets of the set s = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} and checking whether that subset can be represented by the given N diodes. Printing the maximum such subset of the given set which can be represented using N diodes.

Complexity : O(2^|s|) As the |s| value of |s| is very small this solution would pass.

Approach 2: The problem is similar to the 0/1 knapsack problem where the weights of the item are the number of diodes required to represent the number. But the value of the subset which is selected is not the sum of the values of each individual item but the concatenation of the digits in non-increasing order.

Complexity : O(|s| * N) As the value of |s| = 10 a constant it is O(N)

Code of 2nd Approach (C++) :

```
#include<bits/stdc++.h>
using namespace std;
#define DEBUG 0

typedef long long ll;

ll w[] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6};
ll v[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

ll concat(ll dig, ll num){
	ll m = num, d = 0, val;
	if(num < 0)
		return dig;
	if(num == 0)
		d = 1;
	while(m){
		d++;
		m/=10;
	}
	val = (dig*pow(10, d)) + num;
	return val;
}

void kp(int c){
	ll k[12][c+2];
	ll i, j;
	for(i=0; i<=10; i++){
		for(j=0; j<=c; j++){
			if(i==0 || j==0)
				k[i][j] = -1;
			else if(w[i-1] > j)
				k[i][j] = k[i-1][j];
			else
				k[i][j] = max(k[i-1][j], concat(v[i-1], k[i-1][j-w[i-1]]));
		}
	}
	#if DEBUG == 1
		for(i=0; i<=10; i++){
			for(j=0; j<=c; j++){
				printf("%6lld ", k[i][j]);
```

```
                        }
                        cout<<endl;
                }
        #endif
        printf("%lld\n", k[10][c]);
}


int main(){
        int t, n;
        scanf("%d", &t);
        while(t--){
                scanf("%d", &n);
                kp(n);
        }
        return 0;
}
```