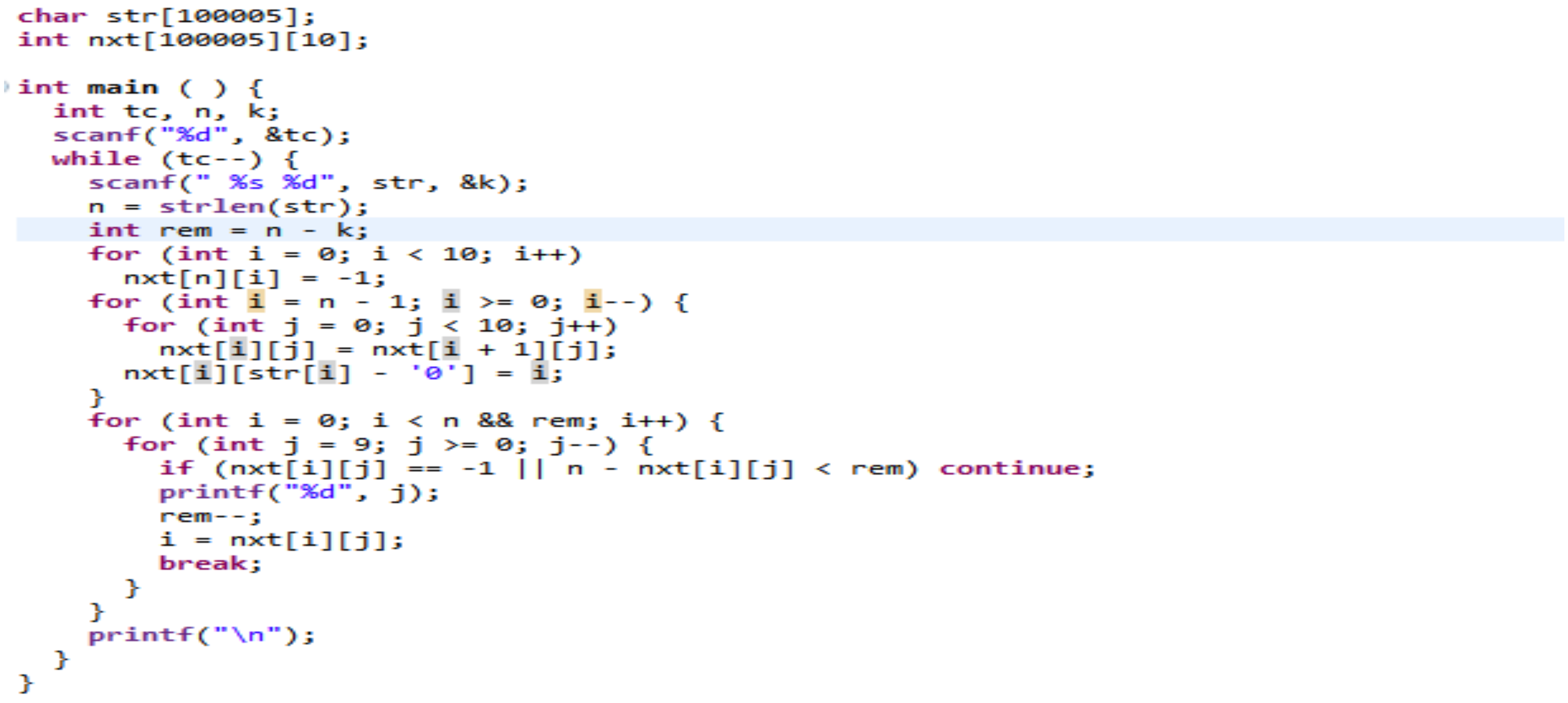
Next Array

spoj.com/problems/MAX_NUM/

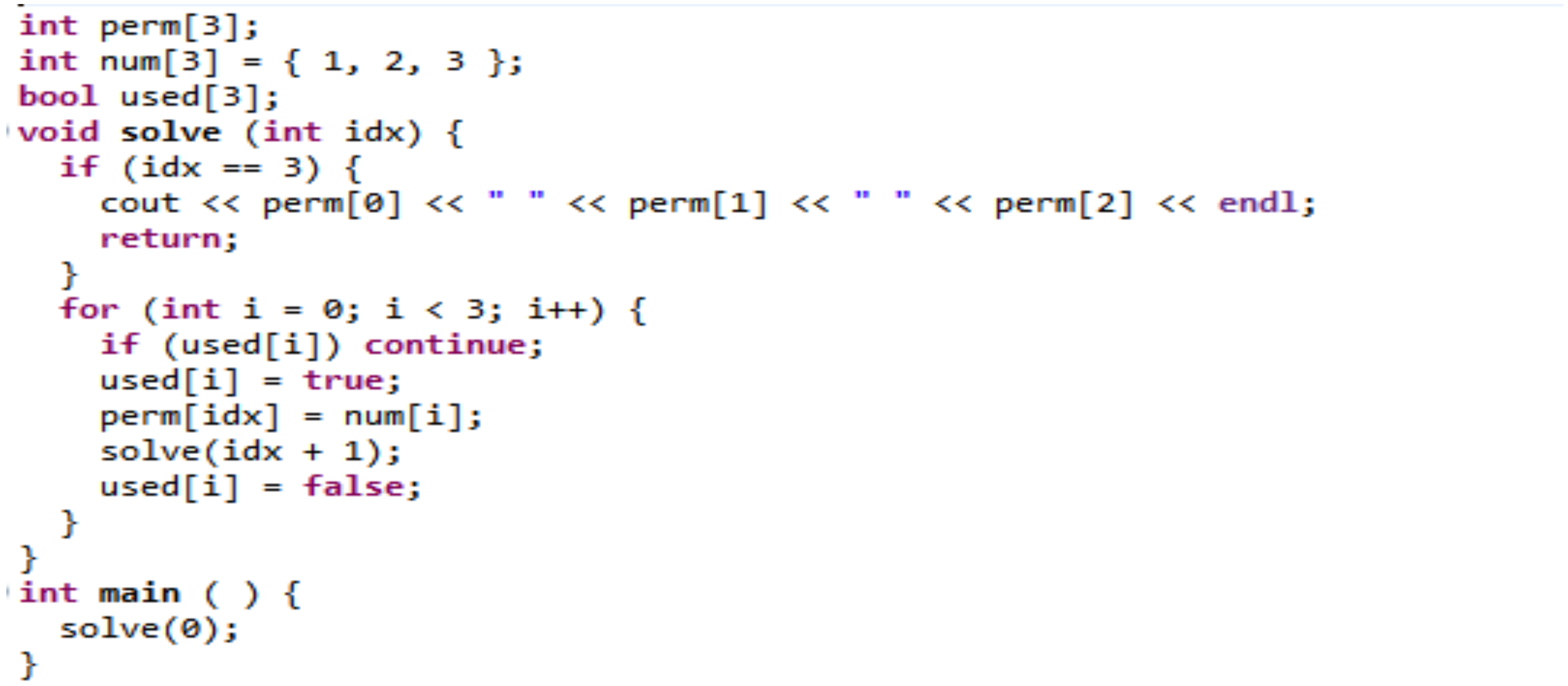


```
char str[100005];
int nxt[100005][10];

int main ( ) {
   int tc, n, k;
   scanf("%d", &tc);
   while (tc--) {
      scanf(" %s %d", str, &k);
      n = strlen(str);
      int rem = n - k;
      for (int i = 0; i < 10; i++)
         nxt[n][i] = -1;
      for (int i = n - 1; i >= 0; i--) {
         for (int j = 0; j < 10; j++)
            nxt[i][j] = nxt[i + 1][j];
         nxt[i][str[i] - '0'] = i;
      }
      for (int i = 0; i < n && rem; i++) {
         for (int j = 9; j >= 0; j--) {
            if (nxt[i][j] == -1 || n - nxt[i][j] < rem) continue;
            printf("%d", j);
            rem--;
            i = nxt[i][j];
            break;
         }
      }
      printf("\n");
   }
}
```

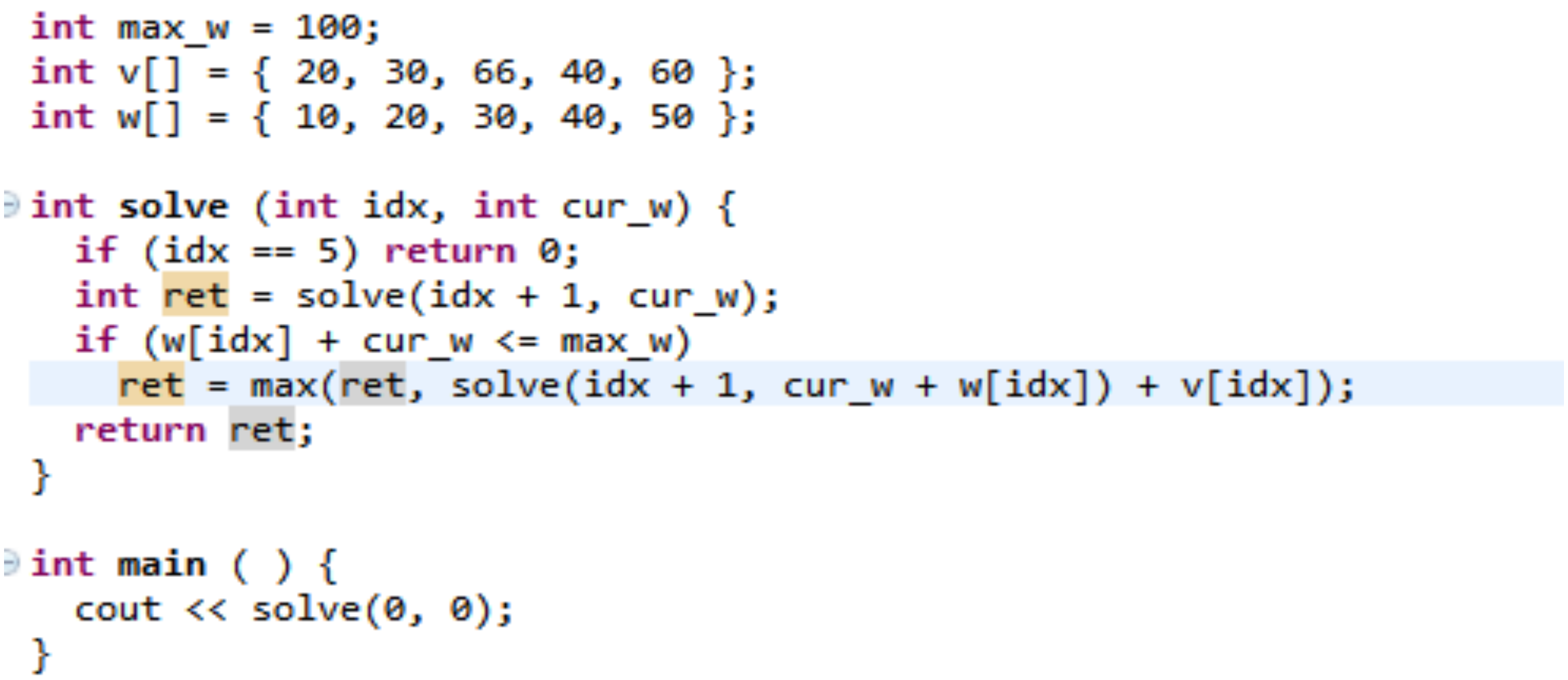
Backtracking

Next Permutation



```
int perm[3];
int num[3] = { 1, 2, 3 };
bool used[3];
void solve (int idx) {
  if (idx == 3) {
    cout << perm[0] << " " << perm[1] << " " << perm[2] << endl;
    return;
  }
  for (int i = 0; i < 3; i++) {
    if (used[i]) continue;
    used[i] = true;
    perm[idx] = num[i];
    solve(idx + 1);
    used[i] = false;
  }
}
int main ( ) {
  solve(0);
}
```

Knapsack

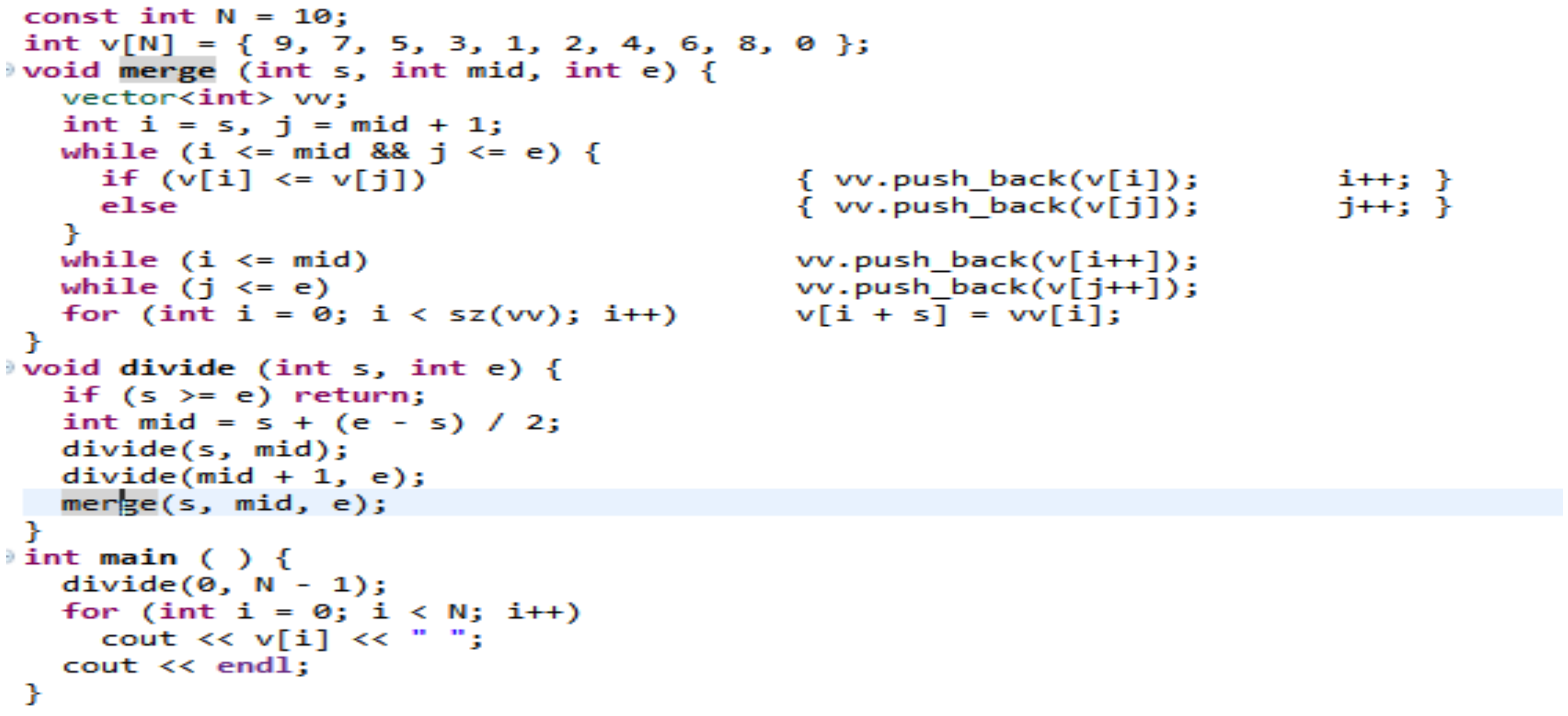


```
int max_w = 100;
int v[] = { 20, 30, 66, 40, 60 };
int w[] = { 10, 20, 30, 40, 50 };

int solve (int idx, int cur_w) {
  if (idx == 5) return 0;
  int ret = solve(idx + 1, cur_w);
  if (w[idx] + cur_w <= max_w)
    ret = max(ret, solve(idx + 1, cur_w + w[idx]) + v[idx]);
  return ret;
}

int main ( ) {
  cout << solve(0, 0);
}
```

D&C - Merge Sort



```
const int N = 10;
int v[N] = { 9, 7, 5, 3, 1, 2, 4, 6, 8, 0 };
void merge (int s, int mid, int e) {
   vector<int> vv;
   int i = s, j = mid + 1;
   while (i <= mid && j <= e) {
      if (v[i] <= v[j])                        { vv.push_back(v[i]);       i++; }
      else                                     { vv.push_back(v[j]);       j++; }
   }
   while (i <= mid)                            vv.push_back(v[i++]);
   while (j <= e)                              vv.push_back(v[j++]);
   for (int i = 0; i < sz(vv); i++)            v[i + s] = vv[i];
}
void divide (int s, int e) {
   if (s >= e) return;
   int mid = s + (e - s) / 2;
   divide(s, mid);
   divide(mid + 1, e);
   merge(s, mid, e);
}
int main ( ) {
   divide(0, N - 1);
   for (int i = 0; i < N; i++)
      cout << v[i] << " ";
   cout << endl;
}
```

Complexity

exponential!!