```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "from itertools import permutations"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[[6, 4, 1, 2, 7], [10, 5, 3, 9], [15, 8, 12], [23, 20], [43]]\n",
      "[[7, 2, 1, 4, 6], [9, 3, 5, 10], [12, 8, 15], [20, 23], [43]]\n",
      "[[6, 1, 3, 2, 8], [7, 4, 5, 10], [11, 9, 15], [20, 24], [44]]\n",
      "[[8, 2, 3, 1, 6], [10, 5, 4, 7], [15, 9, 11], [24, 20], [44]]\n",
      "[[6, 4, 1, 2, 9], [10, 5, 3, 11], [15, 8, 14], [23, 22], [45]]\n",
      "[[9, 2, 1, 4, 6], [11, 3, 5, 10], [14, 8, 15], [22, 23], [45]]\n"
     ]
    }
   ],
   "source": [
    "def pyramid_sum(target, *args):\n",
    "    \"\"\"This function calculates next rows of the tree by summing
adjacent 2 numbers from the previous row.\n",
    "       rolling_lst is set to list of numbers that are being passed
in the function. \n",
    "       If a result of summation of 2 adjacent values is not being
present in rolling_lst then it is added to it.\n",
    "       Once above condition is violated the function returns None.
\n",
    "       If the top hexagon is not equal to target the function also
returns None.\n",
    "       Otherwise, it returns a list of lists as an answer to the
problem\"\"\"\n",
    "    row_lst_base = [_ for _ in args]\n",
    "    rolling_lst = list(args)\n",
    "    row_lst = [row_lst_base]\n",
    "    for level in range(len(args) - 1):\n",
    "        row_lst_new = []\n",
    "        for i in range(len(row_lst[level]) - 1):\n",
    "            pair_sum = row_lst[level][i] + row_lst[level][i + 1]\n",
    "            if pair_sum not in rolling_lst:\n",
    "                rolling_lst += [pair_sum]\n",
    "                row_lst_new.append(pair_sum)\n",
    "            else:\n",
```

```
    "                return None\n",
    "        row_lst += [row_lst_new]\n",
    "    \n",
    "    return row_lst if pair_sum == target else None\n",
    "\n",
    "perm = permutations(range(1, 12), 5)\n",
    "\n",
    "targets = [43, 44, 45]\n",
    "for target in targets:\n",
    "    max_left_element = target - 32\n",
    "    perm = permutations(range(1, 12), 5)\n",
    "    for idx, args in enumerate(list(perm)):\n",
    "        res = []\n",
    "        p_sum = pyramid_sum(target, *args)\n",
    "        if p_sum:\n",
    "            print(p_sum)\n",
    "            res += p_sum"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.7.1"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 4
}
```