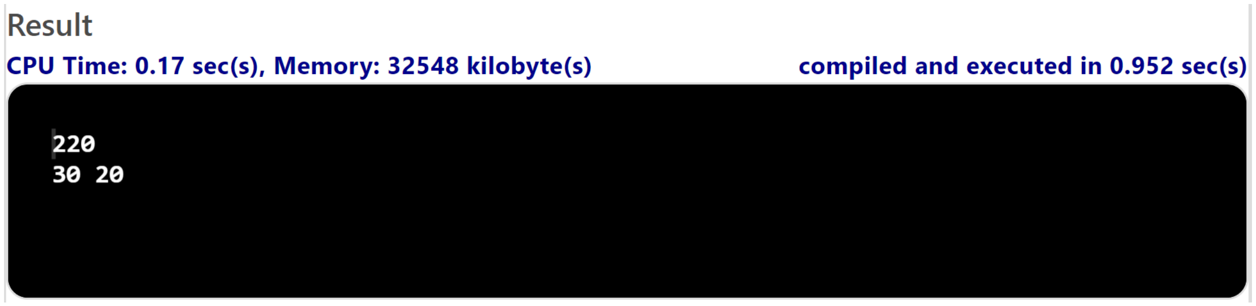
Sanjay Ankola
202010052

Write a program to solve 0/1 knapsack problem using backtracking approach.

```
public class GFG {

        static int max(int a, int b)
        {
                return (a > b) ? a : b;
        }

        static void printknapSack(int W, int wt[],
                                                int val[], int n)
        {
                int i, w;
                int K[][] = new int[n + 1][W + 1];

                for (i = 0; i <= n; i++) {
                        for (w = 0; w <= W; w++) {
                                if (i == 0 || w == 0)
                                        K[i][w] = 0;
                                else if (wt[i - 1] <= w)
                                        K[i][w] = Math.max(val[i - 1] +
                                                K[i - 1][w - wt[i - 1]], K[i - 1][w]);
                                else
                                        K[i][w] = K[i - 1][w];
                        }
                }

                int res = K[n][W];
                System.out.println(res);

                w = W;
                for (i = n; i > 0 && res > 0; i--) {

                        if (res == K[i - 1][w])
                                continue;
                        else {
```

```
                        System.out.print(wt[i - 1] + " ");
                        res = res - val[i - 1];
                        w = w - wt[i - 1];
                    }
                }
            }
            public static void main(String arg[])
            {
                int val[] = { 60, 100, 120 };
                int wt[] = { 10, 20, 30 };
                int W = 50;
                int n = val.length;

                printknapSack(W, wt, val, n);
            }
}
```

Output:



Result

CPU Time: 0.17 sec(s), Memory: 32548 kilobyte(s) compiled and executed in 0.952 sec(s)

```
220
30 20
```