Roll_NO:03

Practical No 04: 0-1 Knapsack problem using dynamic programming or branch and bound

strategy.

#Code:

```
#include <iostream>
#include<vector>
using namespace std;
vector<int>weight={10,20,30};
vector<int>value={60,100,120};
int func(int i,int w){
  if(w==0|| i<0) return 0;


  int ans=func(i-1,w);
  if(w-weight[i]>=0)
  ans=max(ans,func(i-1,w-weight[i])+value[i]);

  return ans;




}
int main()
{
```

```
    int w=50;
    int n=weight.size();

    cout<<"Maximum Profit:"<<func(n,w);


    return 0;
}
```

#output:

Maximum Profit:220