Coin-changing-DA-1.c

```
#include<stdio.h>
void coin_change(int coin[],int totalCoin,int change)
{
   int m[change+1],minimum,i,j;
   m[0]=0;
   for(i=1; i<=change; i++){
      minimum=change+1;
      for(j=0; j<totalCoin; j++){
         if(coin[j]<=i){
               if(m[i-coin[j]]+1 < minimum)
               minimum= m[i-coin[j]]+1;
         }
      }
      m[i]=minimum;
   }
   if(m[change]==0)
      printf("Change is not possible\n");
   else
    printf("Coin need: %d \n",m[change]);
}
   int main() {
   int i,totalCoin=4,change=16;
   int coin[]= {1,2,8,12};
   coin_change(coin,totalCoin,change);
   return 0;
}
```

Coin-changing-DA-2.c

```
#include<stdio.h>
void coin_change(int coin[],int totalCoin,int change)
{
   int m[change+1],minimum,i,j;
   m[0]=0;
   for(i=1; i<=change; i++){
      minimum=change+1;
      for(j=0; j<totalCoin; j++){
         if(coin[j]<=i){
               if(m[i-coin[j]]+1 < minimum)
               minimum= m[i-coin[j]]+1;
         }
      }
      m[i]=minimum;
   }
   if(m[change]==0)
      printf("Change is not possible\n");
   else
     printf("Coin need: %d \n",m[change]);
}
    int main(){
   int i,totalCoin=3,change=16;
   int coin[3]={1,5,10};
   coin_change(coin,totalCoin,change);
  return 0;
}
```

Coin-changing-GA-1.c

```
#include<stdio.h>
void Sort(int ara[],int n)
{
   int i,j,p;
   for(i=0;i<n;i++)
   {
      for(j=0;j<n-1-i;j++)
      {
         if(ara[i]>ara[i-1])
         {
            p=ara[i+1];
            ara[i+1]=ara[i];
            ara[i]=p;
         }
      }
   }
}
void coin_change(int coins[], int n, int m)
{
  int cnt[n],i;
  for(i=0;i<n;i++)cnt[i]=0;
  for(i=n-1;i>=0;i--)
```

```
  {
     if(coins[i]<=m)
     {
        cnt[i]=m/coins[i];
        m=m%coins[i];
     }
  }
  if(m!=0)
      printf("Change is not possible\n");
  else
  {
     printf("Coin need:\n");
     for(i=n-1;i>=0;i--)
     {
        if(cnt[i]!=0)
           printf("%d coin : %d times\n",coins[i],cnt[i]);
     }
  }
}

int main()
{
   int n=4,change=15;
```

```
    int coins[]={1,7,7,10};
    Sort(coins,n);
    coin_change(coins,n,change);
    return 0;
}
```

Coin-chaning-GA-2.c

```
#include<stdio.h>
void Sort(int ara[],int n)
{
    int i,j,p;
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1-i;j++)
        {
            if(ara[i]>ara[i-1])
            {
                p=ara[i+1];
                ara[i+1]=ara[i];
                ara[i]=p;
            }
        }
```

```
    }
}
void coin_change(int coins[], int n, int m)
{
  int cnt[n],i;
  for(i=0;i<n;i++)cnt[i]=0;
  for(i=n-1;i>=0;i--)
  {
     if(coins[i]<=m)
     {
        cnt[i]=m/coins[i];
        m=m%coins[i];
     }
  }
  if(m!=0)
      printf("Change is not possible\n");
  else
  {
     printf("Coin need:\n");
     for(i=n-1;i>=0;i--)
     {
        if(cnt[i]!=0)
           printf("%d coin : %d times\n",coins[i],cnt[i]);
     }
  }
}

int main()
{
```

```
    int n=5,change=12;
    int coins[]={2,5,3,4,6};
    Sort(coins,n);
    coin_change(coins,n,change);
    return 0;
}
```

Fibonacci-num-DA-1.c

```
#include<stdio.h>
int fib(int n)
{
   if (n <= 1)
   return n;
   return fib(n-1) + fib(n-2);
}
 int main ()
{
  int n;
  printf("Enter Any Number : ");
  scanf("%d",&n);
  printf("Fibonacci Number : %d", fib(n));
  getchar();

  return 0;
}
```

Fibonacci-num-DA-2.c

```
#include<stdio.h>

int fib(int n)
{
   int f[n+2],i;
   f[0] = 0;
   f[1] = 1;
   for (i = 2; i <= n; i++){
      f[i] = f[i-1] + f[i-2];
   }
   return f[n];
}
   int main()
{
   int n,t;
   printf("Test Case:");
   scanf("%d",&t);
   for(int i=1;i<=t;i++){
   printf("Number %d:",i);
   scanf("%d",&n);
   printf("Fibonacci %d: %d\n",i,fib(n));
   }

   return 0;
}
```

Fractional-Knapsack-DA-1.c

```
#include <stdio.h>


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


int knapsack(int W, int wt[], int v[], int n)
{
  int i, w;
  int K[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] = max(v[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);
        else
             K[i][w] = K[i-1][w];
     }
  }


  return K[n][W];
}
```

```
int main()
{
   int v[] = {12, 10, 20, 15};
   int wt[] = {2, 1, 3, 2};
   int  W = 5;
   int n = sizeof(v)/sizeof(v[0]);
   printf("Maximum Profit:%d", knapsack(W, wt, v, n));
   return 0;
}
```

Fractional-Knapsack-GA-1.c

```
#include <stdio.h>

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

int knapsack(int W, int wt[], int v[], int n)
{
  int i, w;
  int K[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] = max(v[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);
         else
               K[i][w] = K[i-1][w];
      }

   }

   return K[n][W];

}

   int main()
{
   int v[] = {30, 40, 45, 77, 90};
   int wt[] = {5, 10, 15, 22, 25};
   int  W = 60;
   int n = sizeof(v)/sizeof(v[0]);
   printf("Maximum Profit:%d", knapsack(W, wt, v, n));
   return 0;

}
```

Fractional-knapsack-DA-2.c

```
#include <stdio.h>


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


int knapsack(int W, int wt[], int v[], int n)
{
  int i, w;
  int K[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] = max(v[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);
        else
            K[i][w] = K[i-1][w];
     }
  }


  return K[n][W];
}


   int main()
{
   int v[] = {20, 10, 30};
   int wt[] = {100,50,150};
```

```
    int  W = 50;
    int n = sizeof(v)/sizeof(v[0]);
    printf("Maximum Profit:%d", knapsack(W, wt, v, n));
    return 0;
}
```