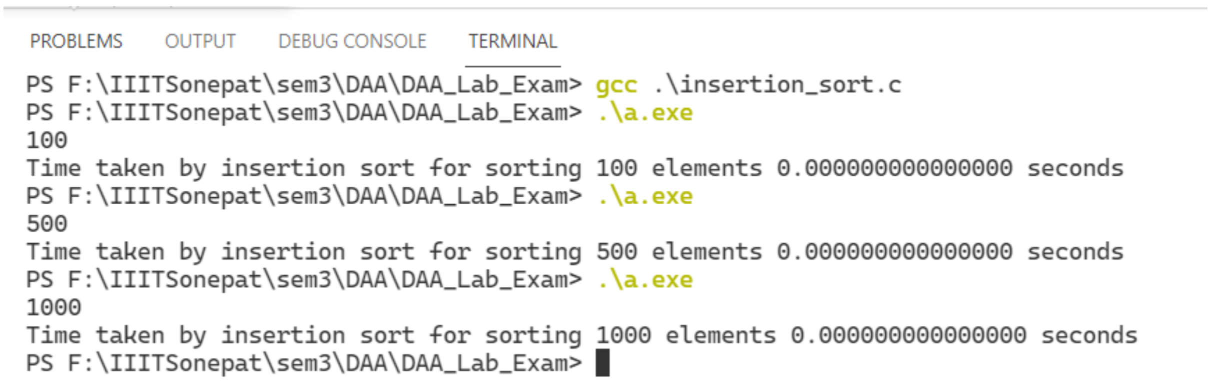
Insertion Sort

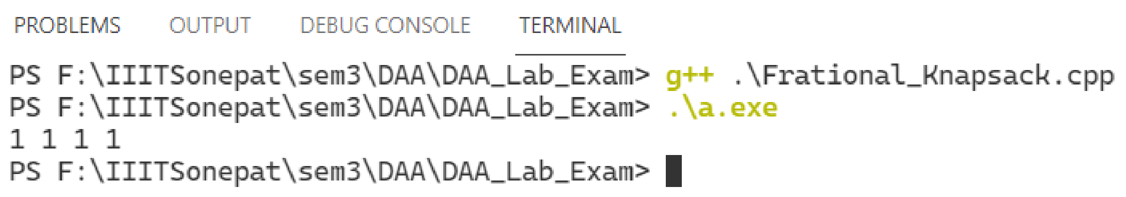
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void insertion_sort(int arr[], int n)
{
   int i, j, key;
   for (i = 1; i < n; i++)
   {
      key = arr[i];
      j = i - 1;
      while (j >= 0 && arr[j] > key)
      {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}

int main()
{
   int n;
   scanf("%d", &n);
   int arr[n];
   for (int i = 0; i < n; i++)
   {
      arr[i] = rand();
   }

   float time = 0.0f;

   clock_t begin = clock();

   insertion_sort(arr, n);

   clock_t end = clock();
   time += (float)(end - begin) / CLOCKS_PER_SEC;

   printf("Time taken by insertion sort for sorting %d elements %.15f
seconds", n, time);

   return 0;
}
```



```
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> gcc .\insertion_sort.c
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> .\a.exe
100
Time taken by insertion sort for sorting 100 elements 0.000000000000000 seconds
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> .\a.exe
500
Time taken by insertion sort for sorting 500 elements 0.000000000000000 seconds
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> .\a.exe
1000
Time taken by insertion sort for sorting 1000 elements 0.000000000000000 seconds
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam>
```

Fractional Knapsack

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
   int p[] = {100, 280, 120, 120};
   int w[] = {10, 40, 20, 24};
   int m = 60;

   double x[4];

   for (int i = 0; i < 4; i++)
   {
      x[i] = 0;
   }

   int weight = 0;

   for (int i = 0; i < 4; i++)
   {
      if (weight + w[i] <= m)
      {
         x[i] = 1;
      }
      else
      {
         x[i] = (m - weight) / w[i];
         weight = m;
         break;
      }
   }

   for (int i = 0; i < 4; i++)
   {
      cout << x[i] << " ";
   }

   return 0;
}
```

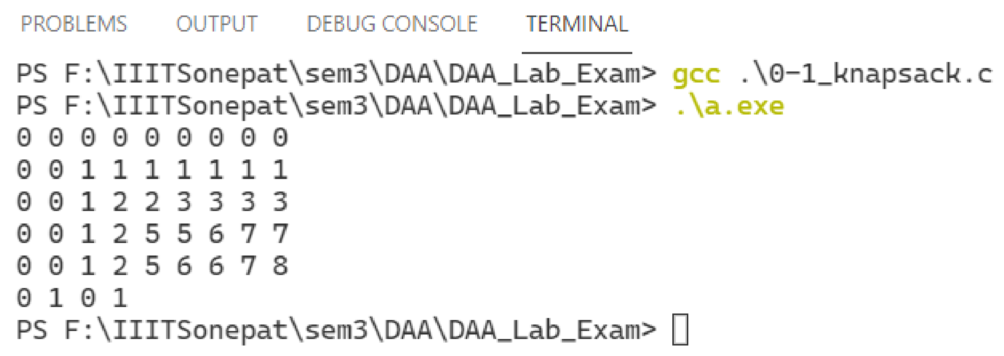


```
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> g++ .\Frational_Knapsack.cpp
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> .\a.exe
1 1 1 1
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> 
```

0/1 Knapsack

```c
#include <stdio.h>
int max(int a, int b)
{
   if (a > b)
      return a;
   else
      return b;
}
int main()
{
   int m = 8;
   int p[] = {0, 1, 2, 5, 6};
   int w[] = {0, 2, 3, 4, 5};
   int n = 4;
   int arr[n + 1][m + 1];

   for (int i = 0; i <= n; i++)
   {
      for (int j = 0; j <= m; j++)
      {
         if (i == 0 || j == 0)
         {
            arr[i][j] = 0;
         }

         else if (w[i] <= j)
         {
            arr[i][j] = max(arr[i - 1][j], arr[i - 1][j - w[i]] + p[i]);
         }
         else
         {
            arr[i][j] = arr[i - 1][j];
         }
      }
   }

   for (int i = 0; i <= n; i++)
   {
      for (int j = 0; j <= m; j++)
      {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }

   int result[n];
```

```c
    int i = n;
    int j = m;
    while (i >= 0 && j >= 0)
    {
        if (arr[i][j] == arr[i - 1][j])
        {
            result[i - 1] = 0;
            i--;
        }
        else
        {
            result[i - 1] = 1;
            j = j - w[i];
            i--;
        }
    }
    for (int i = 0; i < n; i++)
    {
        printf("%d ", result[i]);
    }

    return 0;
}
```



```
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> gcc .\0-1_knapsack.c
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> .\a.exe
0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1
0 0 1 2 2 3 3 3 3
0 0 1 2 5 5 6 7 7
0 0 1 2 5 6 6 7 8
0 1 0 1
PS F:\IIITSonepat\sem3\DAA\DAA_Lab_Exam> 
```