Q1) GCD

```
#include<stdio.h>
void main() {
   int a, b, m, n, r, modcount = 0, assigncount = 0;

   printf("Key in the two numbers [first number should be greater than second number ] whose GCD is to be cacalculated\n");

   scanf("%d %d",&a, &b);
   m = a;
   n = b;

   while (n > 0) {
      r = m % n;
      modcount++;
      m = n;
      n = r;
      assigncount = assigncount + 2;
   }

   printf("\n\tGCD of %d and %d is %d. \n\tThe mod function was called %d time[s].\n\tAssignment operation was done %d times \n", a, b, m, modcount,assigncount);
}
```

```
/tmp/ZazlfPwaUL.o
Key in the two numbers [first number should be greater than second number ]
    whose GCD is to be cacalculated
15265
15

    GCD of 15265 and 15 is 5.
    The mod function was called 3 time[s].
    Assignment operation was done 6 times
```

Q2) Comparison of Horner's rule and brute force method

```
#include<stdio.h>
void main()
int P, x, degree, coeff[10], i, loopmultaddcount=0; int p[10], k, pbrute=0, brutemultcount=0;
printf("Key in the degree of the polynomialin");
scan("d", &degree);
printf("Key in the %d coefficients starting from that for
degree %din", degree + 1, degree);
for (i = 0; i <= degree; i++)
scanf("%d",
', &coeff[i]);
printf("the coefficients aren");
for (i = 0; i <= degree; i++)
printf("@dit", coeff[i] );
printf("n"):
printf("Key in the value for Xin");
scanf("d", &x);
//Horner's method
P=coeff[0];
for i = 1; i <= degree; i++)
P = P*x + coeff[i];
loopmultaddcount++;
//Brute force method
for (k = 0; k <= degree; k++)
{
p[k]=coeff[k];
for(i=1; i<= degree-k; i++)
p[k] = p[k] * x;
brutemultcount++;
}
for (i=0; i <= degree; i++)
pbrute = pbrute + p[l;
```

printf("In\tP(x) at x = %d is P(%d) = %d. In\tIn the

Horner's method InItit[1] The loop was iterated %d time[s].\ n|t\[2] There were d multiplication operations.In\t\t[3]

There were %d addition operations.\n",

', x, X, P,

loopmultaddcount, loopmultaddcount, loopmultaddcount);

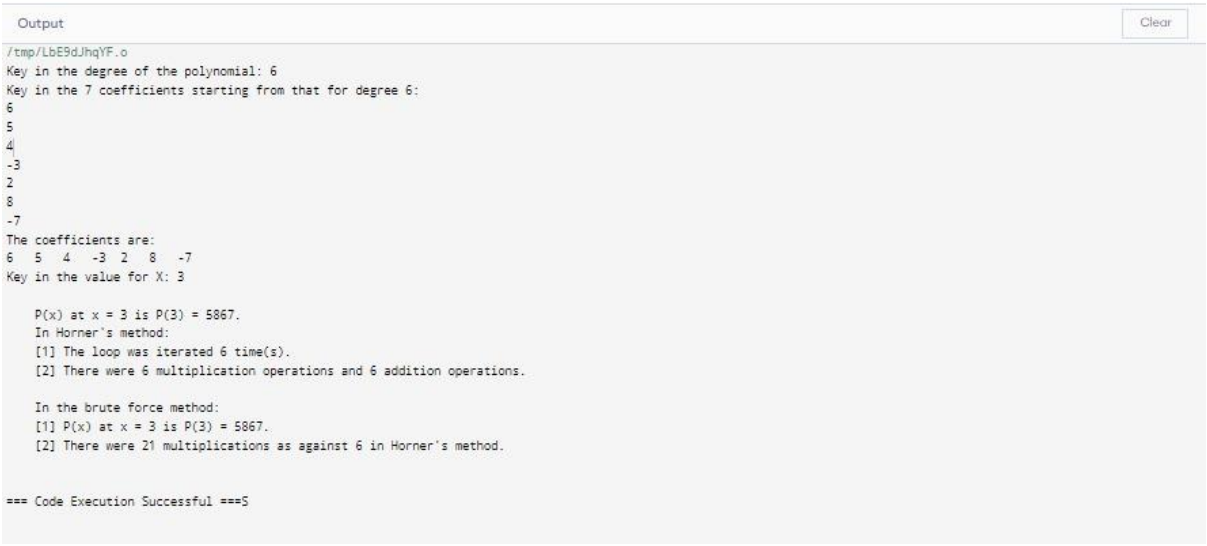
printf"In\tin the brute force method\n\t\t[1] P(x) at x =

%d is P(%d) = %d. In\t\t[2] There were %d multiplications as

against %d in the Horner's method. In\n" , x, x, pbrute,

brutemultcount, loopmultaddcount);

}



Q 3) Matrix Multiplication

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

void main() {
   int A[10][10], B[10][10], C[10][10];
   int rA, cA, rB, cB;
   int i, j, k;
   int outerloopcount = 0, middleloopcount = 0, innerloopcount = 0, addncount = 0, multcount
= 0;

   printf("Key in the row size and column size [maximum is 10 X 10] for the first matrix, say,
A\n");
   scanf("%d %d", &rA, &cA);

   printf("Key in the row size and column size [maximum is 10 X 10] for the second matrix,
say, B\n");
   scanf("%d %d", &rB, &cB);

   if (cA != rB) {
      printf("\nMatrices are incompatible for multiplication\n");
      exit(0);
   }

   printf("Key in row-wise the elements of the first [%d X %d] matrix A\n", rA, cA);
   for (i = 0; i < rA; i++)
      for (j = 0; j < cA; j++)
         scanf("%d", &A[i][j]);

   printf("\nFirst matrix A is:\n");
   for (i = 0; i < rA; i++) {
      for (j = 0; j < cA; j++)
         printf("\t%d\t", A[i][j]);
      printf("\n");
   }

   printf("\nKey in row-wise the elements of the second [%d X %d] matrix B\n", rB, cB);
   for (i = 0; i < rB; i++)
      for (j = 0; j < cB; j++)
         scanf("%d", &B[i][j]);

   printf("\nSecond matrix B is:\n");
   for (i = 0; i < rB; i++) {
      for (j = 0; j < cB; j++)
         printf("\t%d\t", B[i][j]);
      printf("\n");
   }

   // Matrix multiplication
   for (i = 0; i < rA; i++) {
      outerloopcount++;
```

```
        for (j = 0; j < cB; j++) {
            middleloopcount++;
            C[i][j] = 0;
            for (k = 0; k < cA; k++) {
                innerloopcount++;
                C[i][j] += A[i][k] * B[k][j]; // Use * for multiplication

                addncount++;
                multcount++;
            }
        }
    }

    printf("\nThe matrix product is C = A X B and C is:\n");
    for (i = 0; i < rA; i++) {
        for (j = 0; j < cB; j++)
            printf("\t%d\t", C[i][j]);
        printf("\n");
    }

    printf("\nStatistics:\n");
    printf("[1] Outer loop was iterated for %d time(s).\n", outerloopcount);
    printf("[2] Middle loop was iterated for %d time(s).\n", middleloopcount);
    printf("[3] Inner loop was iterated for %d time(s).\n", innerloopcount);
    printf("[4] %d additions were done.\n", addncount);
    printf("[5] %d multiplications were done.\n", multcount);
}
```

Output Clear

```
/tmp/NkOlLqNUVv.o
Key in the row size and column size [maximum is 10 X 10] for the first
    matrix, say, A
5 5
Key in the row size and column size [maximum is 10 X 10] for the second
    matrix, say, B
5 2
Key in row-wise the elements of the first [5 X 5] matrix A
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5

First matrix A is:
    1       2       3       4       5
    1       2       3       4       5
    1       2       3       4       5
    1       2       3       4       5
    1       2       3       4       5
```

```
Key in row-wise the elements of the second [5 X 2] matrix B
1 10
2 10
3 10
4 10
5 10

Second matrix B is:
    1       10
    2       10
    3       10
    4       10
    5       10

The matrix product is C = A X B and C is:
    55      150
    55      150
    55      150
    55      150
    55      150

Statistics:
[1] Outer loop was iterated for 5 time(s).
[2] Middle loop was iterated for 10 time(s).
[3] Inner loop was iterated for 50 time(s).
[4] 50 additions were done.
[5] 50 multiplications were done.
```

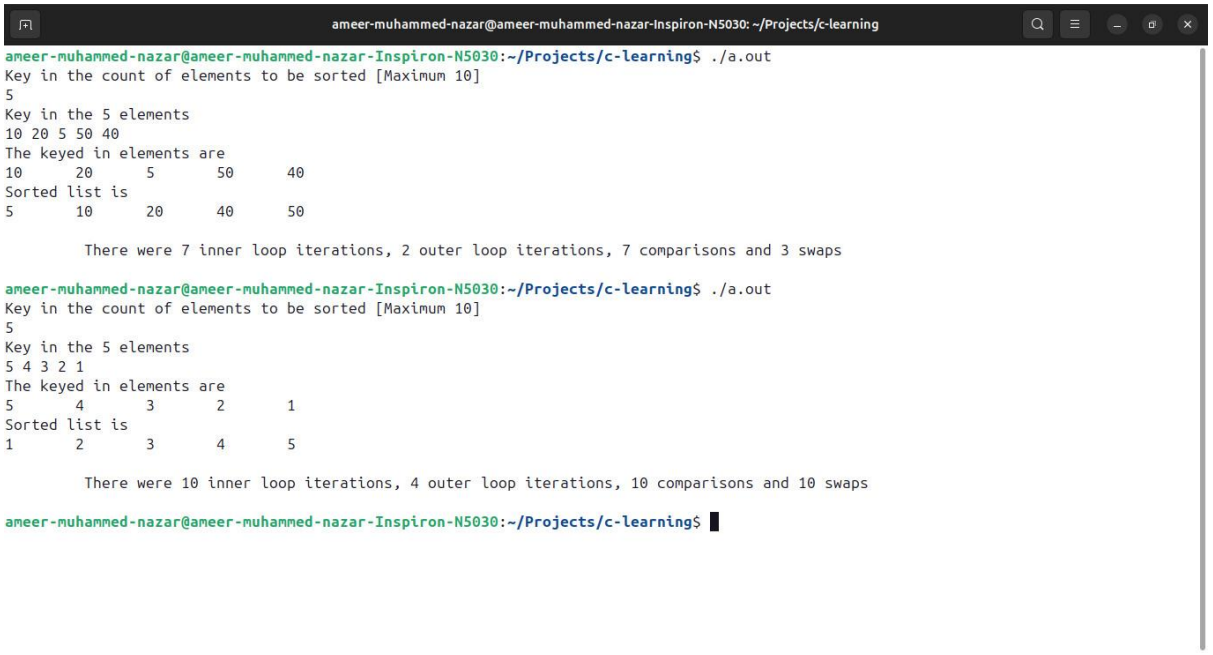
Q4) Left to Right Binary Exponentiation

```
#include <stdio.h>
#include <string.h>
int binaryexp(float base, int power)
{
   int binrev[100], bin[100], i, l, j, k, powers;
   double result;
   i++;
   powers = power;
   i = 0;
   while (power)
   {
      binrev[i] = power % 2;
      power = power / 2;
      i++;
      l = i;
   }
   for (k = 0; k < l; k++, i--)
      bin[k] = binrev[i - 1];
   printf("\n binary is \n");
   for (k = 0; k < l; k++)
      printf("\t%d \t", bin[k]);
   printf("\n");
   result = 1;
   for (j = 0; j < l; j++)
   {
      if (bin[j] == 1)
         result = result * result * base;
      if (bin[j] == 0)
         result = result * result;
   }
   printf("\n %f to the power %d using left to right binary exponentiation is \n\n\t%lf \n\n", base, powers, result);
}
void main()
{
   float x;

   int n;

   printf("\nKey in the base number X whose power is to be calculated using left to right binary exponentiation is
required and the value of power / index \n\n");

   scanf("%f%d", &x, &n);

   printf("\n------X = %f..... power / index = %d\n", x, n);

   binaryexp(x, n);

}
```

```
PS D:\My_Documents\IGNOU\Lab\MCS216\Q4>  & 'c:\Users\Sreek\.vscode\extensions\ms-vscode.cpptools-1.21.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe'
 '--stdin=Microsoft-MIEngine-In-dspi3tgq.fs3' '--stdout=Microsoft-MIEngine-Out-hks0rbvm.ark' '--stderr=Microsoft-MIEngine-Error-wvidjkzb.idt' '--pid=Microsof
t-MIEngine-Pid-b3d5lrnn.4gg' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'

Key in the base number X whose power is to be calculated using left to right binary exponentiation is required and the value of power / index

3. 31

------X = 3.000000..... power / index = 31

 binary is
       1             1             1             1             1

 3.000000 to the power 31 using left to right binary exponentiation is

        617673396283947.000000
```

Q5) Bubble sort

```
#include <stdio.h>

void main(){

int A[10], i, j, n, t, cmpcount = 0 , swapcount=0,outerloopcount = 0 , innerloopcount=0,
swapcheck=0;printf("Key in the count of elements to be sorted

[Maximum 10] \n");

scanf("%d", &n);

printf("Key in the %d elements\n", n);

for ( i = 0 i < n i++)

scanf("%d", &A[i]);

printf("\nThe keyed in elements are \n");for ( i = 0 i < n i++)

printf("%d\t", A[i]);

printf("\n");

for ( i = 0 i <= n - 2 i++){

if (i!= 0 && swapcheck == 0) break;outerloopcount++;

for ( j = 0 j <= n - j - 2 j ++){

innerloopcount++;

cmpcount++;

swapcheck = 0;if (A[j] > A[j + 1]){

swapcheck = 1;swapcount++;

t = A[j];

A[j] = A[j+1];

A[j+1] = t;}

}}

printf("\nthe sorted list is\n");

for (i=0; i <= n-1; i++)

printf("\t%d", A[i]);

printf("\n");

printf("\n\tThere were %d inner loop iterations, %d outer loop iterations, %d comparisions and %d
swaps\n\n", innerloopcount, outerloopcount, cmpcount, swapcount);

 }
```


ameer-muhammed-nazar@ameer-muhammed-nazar-Inspiron-N5030: ~/Projects/c-learning
ameer-muhammed-nazar@ameer-muhammed-nazar-Inspiron-N5030:~/Projects/c-learning$ ./a.out
Key in the count of elements to be sorted [Maximum 10]
5
Key in the 5 elements
10 20 5 50 40
The keyed in elements are
10 20 5 50 40
Sorted list is
5 10 20 40 50
There were 7 inner loop iterations, 2 outer loop iterations, 7 comparisons and 3 swaps
ameer-muhammed-nazar@ameer-muhammed-nazar-Inspiron-N5030:~/Projects/c-learning$ ./a.out
Key in the count of elements to be sorted [Maximum 10]
5
Key in the 5 elements
5 4 3 2 1
The keyed in elements are
5 4 3 2 1
Sorted list is
1 2 3 4 5
There were 10 inner loop iterations, 4 outer loop iterations, 10 comparisons and 10 swaps
ameer-muhammed-nazar@ameer-muhammed-nazar-Inspiron-N5030:~/Projects/c-learning$

Q6) Selection Sort

```
#include <stdio.h>

// Function to swap two elements
void swap(int *f, int *s) {
	int t = *f;
	*f = *s;
	*s = t;
}

void main() {
	int A[10], i, j, n;
	int cmpcount = 0, swapcount = 0;
	int outerloopcount = 0, innerloopcount = 0, minpos;

	// Input number of elements
	printf("Key in the count of elements to be sorted [Maximum 10]\n");
	scanf("%d", &n);

	// Input array elements
	printf("Key in the %d elements\n", n);
	for (i = 0; i < n; i++) {
	scanf("%d", &A[i]);
	}

	// Display the input array
	printf("\nThe keyed in elements are \n");
	for (i = 0; i < n; i++) {
	printf("%d\t", A[i]);
	}
	printf("\n");

	// Selection sort algorithm
	for (i = 0; i < n - 1; i++) {
	outerloopcount++;
	minpos = i;
	for (j = i + 1; j < n; j++) {
	innerloopcount++;
	cmpcount++;
	if (A[j] < A[minpos]) {
		minpos = j;
	}
	}
	// Swap if needed
	if (minpos != i) {
	swapcount++;
```

```
        swap(&A[minpos], &A[i]);
        }
        }

        // Display the sorted array
        printf("\nThe sorted list is\n");
        for (i = 0; i < n; i++) {
        printf("\t%d", A[i]);
        }
        printf("\n");

        // Display counts
        printf("\n\tThere were %d inner loop iterations, %d outer loop iterations, %d
comparisons, and %d swaps\n\n",
        innerloopcount, outerloopcount, cmpcount, swapcount);
}
```

Output Clear

```
/tmp/kj6LJZFaXw.o
Key in the count of elements to be sorted [Maximum 10]
5
Key in the 5 elements
50
40
20
10
5

The keyed in elements are
50  40  20  10  5

The sorted list is
    5   10  20  40  50

    There were 10 inner loop iterations, 4 outer loop iterations, 10 comparisons, and 2
        swaps



=== Code Exited With Errors ===
```

Q7) Fractional Knapsack problem

```
#include <stdio.h>

struct item
{
   int id;
   int w;
   int p;
   float value;
};

void main()
{
   int i, j, w, p, tw = 0, capacity, itemcount;
   float tv = 0, partp;
   struct item K[50];
   printf("\nKey in count of items [maximum 50] and the maximum capacity of the bag\n\t");
   scanf("%d%d", &itemcount, &capacity);
   printf("Key in, row-wise [one line per item ], the serial number, the weight and the profit for each of the %d
items\n", itemcount);
   for (i = 0; i < itemcount; i++)
      scanf("%d%d%d", &K[i].id, &K[i].w, &K[i].p);

   for (i = 0; i < itemcount; i++)
      K[i].value = (float)K[i].p / K[i].w;

   struct item KK;

   for (i = 0; i < itemcount - 1; i++)
   {
      for (j = 0; j < itemcount - 1 - i; j++)
      {
         if (K[j + 1].value > K[j].value)
         {
            KK = K[j + 1];
            K[j + 1] = K[j];
            K[j] = KK;
         }
      }
   }
   printf("The %d items arranged in non-descending order of the ratio Value = [ Profit / Weight] is as under\n",
itemcount);

   printf("\n\tValue\tItem serial number\tWeight\tProfit\n");
   for (i = 0; i < itemcount; i++)
      printf("\n\t%0.2f \t\t %d \t\t %d \t %d \n", K[i].value, K[i].id, K[i].w, K[i].p);

   printf("\n");

   printf("The solution to the Fractional Knapsack problem\n\n");
   for (i = 0; i < itemcount; i++)
   {
```

```
    if (K[i].w + tw <= capacity)
    {
      tw += K[i].w;
      tv += K[i].p;
      printf("\nSelected Item %d [whole]\t\tWeight %d \t Profit %d \t\tCumulative Weight\t%d\tCumulative
Value \t%0.2f\n\n", K[i].id, K[i].w, K[i].p, tw, tv);
    }
    else
    {
      w = capacity - tw;
      partp = (float)w * (float)K[i].p / (float)K[i].w;
      tw += w;
      tv += partp;
      printf("\nSelected Item %d [part]\t\tWeight %d \t Profit %0.2f \t\tCumulative Weight\t%d\tCumulative
Value \t%0.2f\n\n", K[i].id, w, partp, tw, tv);
      break;
    }
  }

  printf("\nThus the Knapsack with a capacity of %d can hold items worth a Cumulative Total Value of
\t%0.2f\n\n", tw, tv);

  printf("\n");
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDB SQL CONSOLE cppdbg: main.exe +

```
Key in count of items [maximum 50] and the maximum capacity of the bag
        5 20
Key in, row-wise [one line per item ], the serial number, the weight and the profit for each of the 5 items
1 5 20
2 8 30
3 10 40
4 12 32
5 15 55
The 5 items arranged in non-descending order of the ratio Value = [ Profit / Weight] is as under

        Value   Item serial number      Weight  Profit

        4.00            1               5       20

        4.00            3               10      40

        3.75            2               8       30

        3.67            5               15      55

        2.67            4               12      32

The solution to the Fractional Knapsack problem

Selected Item 1 [whole]         Weight 5        Profit 20           Cumulative Weight       5       Cumulative Value        20.00

Selected Item 3 [whole]         Weight 10       Profit 40           Cumulative Weight       15      Cumulative Value        60.00

Selected Item 2 [part]          Weight 5        Profit 18.75        Cumulative Weight       20      Cumulative Value        78.75

Thus the Knapsack with a capacity of 20 can hold items worth a Cumulative Total Value of       78.75
```