Name: Rahul Ranjan
Roll No: 20CS8016

[CSS352]

Assignment 2

1. Write a C program that takes two integer values as input, displays its sum. The integer value may be so large such that it exceeds the range of the max value of any integer data types.
   Hint: You may store the integer input as a string.

2. Write a C program that takes two integer values as input, displays its product. The integer value may be so large such that it exceeds the range of the max value of any integer data types.
   Hint: You may store the integer input as a string. You may use the large addition function from the previous assignment. [OPTIONAL].

3. Write a C program that takes an input unindented C code file, reads its contents and write the indented code back as the same file name. Try using command line arguments to input the file name. [OPTIONAL]

4. Write a C program to perform matrix multiplication. Use dynamic allocation 2-D arrays to store the matrices. Use malloc and free and make sure you explicitly clear the garbage after processing is done.

5. Write a C program to find the determinant of a matrix. Use dynamic allocation 2-D arrays to store the matrix.
   Hint: Try to write a recursive function for computing determinant of a matrix

6. Write a C program to store the CGPA obtained by the students of different departments. Note that Institute is having a fixed number of Departments (i.e. the number of Depts are known prior to execution) but the number of students in each Dept is known in runtime. Moreover different Dept. have different student capacity. Compute the highest CGPA obtained in each Dept. and the highest CGPA among all Departments.
   Hint: Using dynamic allocation, use malloc and free and make sure you explicitly clear the garbage after processing is done.

Answer 1.

Addition - [Code]

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#define MAX 50

/*
Roll No: 20CS8016

Q1: Write a C program that takes two integer values as input, displays its sum.
[The integer value may be so large such that it exceeds the range of max value of integer type.]
Hint: you may store the integer input as a string
*/

const char * addition(char string_a[MAX], char string_B[MAX]){

  char string_A[MAX] = {'0'};
  strcat(string_A, string_a);

  int no_of_zeroes = abs(strlen(string_A) - strlen(string_B));
  // printf("%d\n", no_of_zeroes);

  char string_temp[MAX];
  for (int i = 0; i < no_of_zeroes; i++){
    string_temp[i] = '0';
  }
  string_temp[no_of_zeroes] = '\0';

  strlen(string_A) > strlen(string_B) ?
  strcat(string_temp, string_B) : strcat(string_temp, string_A);

  strlen(string_A) > strlen(string_B) ?
  sprintf(string_B, "%s" , string_temp) : sprintf(string_A, "%s" , string_temp);
  // printf("A:%s B:%s\n", string_A, string_B);

  static char result[MAX]; int carry = 0;

  for (int i = strlen(string_A) - 1; i != -1; i--){

    if (string_A[i] + string_B[i] + carry - 48 > 57){
      result[i] = (int)string_A[i] + (int)string_B[i] + carry - 58;
      carry = 1;
    } else {
      result[i] = (int)string_A[i] + (int)string_B[i] + carry - 48;
      carry = 0;
    }
    // printf("%c\n", result[i]);
  }

  result[strlen(string_A)] = '\0';

  if (result[0] == 48){
    for (int i = 0; i < strlen(result); i++){
      result[i] = result[i+1];
    }
```

```c

    result[strlen(string_A) - 1] = '\0';
  };

  return result;
}


int main(){

  // Enter values in terminal or clear comment on line 69 for large values in input.txt.

  char string_A[MAX];
  char string_B[MAX];

  // freopen("input.txt", "r", stdin);

  printf("Enter first number: ");
  scanf("%s", string_A);
  printf("Enter second number: ");
  scanf("%s", string_B);
  printf("Check output.txt\n");

  freopen("output.txt", "w", stdout);

  printf("Sum: %s\n\n", addition(string_A, string_B));

  return 0;
}
```

>> Input

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Submissions\Assignment 2> cd
 "d:\Classes\Third Semester\DSA Laboratory Assignments\Submissions\Assignment 2\" ;
if ($?) { gcc question1.c -o question1 } ; if ($?) { .\question1 }
Enter first number: 100000000000000000000
Enter second number: 201030
Check output.txt
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Submissions\Assignment 2>
```

<< Output

```
output.txt

Sum: 100000000000000201030
```

Answer 2

Product - [Code]

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#define MAX 700

/*
Roll No: 20CS8016

Q2: Write a C program that takes two integer values as input, displays its product.
[The integer value may be so large such that it exceeds range of max value of any integer.]
Hint: you may store the integer input as a string.
*/

const char * multiplication(char A[MAX], char B[MAX]){

  char string_A[MAX];
  memset(string_A, '\0', MAX);

  static char result[MAX];
  memset(result, '\0', MAX); //Resets value of static variable;

  //Starting Zero Cleanup --start
  int ind;
  ind = -1;
  for (int i = 0; i < MAX; i++) {
    if (A[i] != '0') {
      ind = i; break;
    }
  }

  for (int i = 0; i < MAX - ind; i++){
    string_A[i] = A[ind + i];
  }

  char string_B[MAX];
  memset(string_B, '\0', MAX);

  ind = -1;
  for (int i = 0; i < MAX; i++) {
    if (B[i] != '0') {
      ind = i; break;
    }
  }

  for (int i = 0; i < MAX - ind; i++){
    string_B[i] = B[ind + i];
  }
  //Starting Zero Cleanup --end

  // printf("A: %s B: %s\n", string_A, string_B);

  int i_n1 = 0;
  int i_n2 = 0;

  for (int i = strlen(string_A) - 1; i >= 0; i--){
    int carry = 0;
    int n1 = string_A[i] - '0';


    i_n2 = 0;
```

```c

    for (int j = strlen(string_B) - 1; j >= 0; j--){
      int n2 = string_B[j] - '0';
      // printf("Previous Carry: %d\n", carry);

      int sum;
      if (result[i_n1 + i_n2])
        sum = n1 * n2 + (result[i_n1 + i_n2] - 48) + carry;
      else
        sum = n1 * n2 + carry;

      carry = sum / 10;

      result[i_n1 + i_n2] = sum % 10 + 48;

      i_n2++;
    }

    if (carry > 0)
      result[i_n1 + i_n2] += carry + 48;

    i_n1++;
  }

  //Reversing the result array
  int j = strlen(result) - 1; int i = 0; int _temp;
  while (i < j){
    _temp = result[i];
    result[i] = result[j];
    result[j] = _temp;
    i++; j--;
  }

  return result;
}


int main(){

  /*
  300! goes as long as 614 digits. So taken upper limit as 700. Highest factorial call can go upto 333!
  Enter the values in terminal or clear comment on line 149 for very large values in input.txt.
  */

  char string_A[MAX];
  char string_B[MAX];

  // freopen("input.txt", "r", stdin);

  printf("Enter first number: ");
  scanf("%s", string_A);
  printf("Enter second number: ");
  scanf("%s", string_B);
  printf("Check output.txt\n");


  freopen("output.txt", "w", stdout);

  printf("Product: %s\n\n", multiplication(string_A, string_B));
  // printf("Factorial of %s:\n%s", string_A, factorial(string_A));
  // printf("Factorial of %s:\n%s", string_B, factorial(string_B));

  return 0;
}
```

>> Input

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Submissions\Assignment 2> cd
 "d:\Classes\Third Semester\DSA Laboratory Assignments\Submissions\Assignment 2\" ;
if ($?) { gcc question2.c -o question2 } ; if ($?) { .\question2 }
Enter first number: 1000000000000000000000
Enter second number: 30000
Check output.txt
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Submissions\Assignment 2>
```

<< Output

output.txt

```
Product: 30000000000000000000000000
```

Answer 3.

Indentation - [Code]

```c
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#define BUFFER_SIZE 1000

/*
Roll No: 20CS8016

Q3: Write a C program that takes an input unindented C code file,
reads its contents and write the indented code back as the same file name.
[Try using command line arguments to input the file name]
*/

int main(int argc, char *argv[]){

  for (int file = 0; file < argc; file++){
    FILE *reader, *writer;
    reader = fopen(argv[file],"r");
    writer = fopen("indented.c","w");

    char readBuffer[BUFFER_SIZE]; char writeBuffer[BUFFER_SIZE];
    memset(readBuffer, '\0', BUFFER_SIZE);

    int indent_level = 0;

    while (fgets(readBuffer, BUFFER_SIZE, reader)){

      memset(writeBuffer, '\0', BUFFER_SIZE);
      bool blockStart = 0;
      for (int i = 0; i < BUFFER_SIZE; i++){
        if (readBuffer[i] == '{'){
          indent_level++;
          blockStart = 1;
          break;
        }
        else if (readBuffer[i] == '}'){
          indent_level--;
          break;
        }
      }

      if (blockStart){
        for (int i = 0; i < indent_level - 1; i++){
          writeBuffer[i] = '\t';
        }
      }
      else {
        for (int i = 0; i < indent_level; i++){
          writeBuffer[i] = '\t';
        }
      }
      strcat(writeBuffer, readBuffer);
      fprintf(writer, writeBuffer);

    }
    fclose(reader);
    fclose(writer);
  }

  return 0;
}
```

>> Input

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Indentation> gcc .\indentation.c -o main
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Indentation> ./main program.c
```

program.c

```c
#include <stdio.h>
int main(){
printf("Start Line");
if ("1" == 1){
printf("True");
printf("True Again");
if (1){
printf("Inside check Statement");
}
else
printf("Inside checks but in else");
}
for (int i = 0; i < 20; i++){
for (int j = i; j < 15; j++){
for (int k = i + j; k < 100; k++){
int something = 0;
something++;
printf("Inside Nested Loop Line 1");
printf("Inside Nested Loop Line 1");
printf("Inside Nested Loop Line 1");
printf("Inside Nested Loop Line 1");
}
}
}
printf("Hello World");
printf("End Line");
return 0;
}
```

<< Output

indented.c

```c
#include <stdio.h>
int main(){
  printf("Start Line");
  if ("1" == 1){
    printf("True");
    printf("True Again");
    if (1){
      printf("Inside check Statement");
    }
    else
    printf("Inside checks but in else");
  }
  for (int i = 0; i < 20; i++){
    for (int j = i; j < 15; j++){
      for (int k = i + j; k < 100; k++){
        int something = 0;
          something++;
            printf("Inside Nested Loop Line 1");
            printf("Inside Nested Loop Line 1");
            printf("Inside Nested Loop Line 1");
            printf("Inside Nested Loop Line 1");
          }
        }
      }
      printf("Hello World");
      printf("End Line");
      return 0;
    }
```

Answer 4.

Matrix Multiplication - [Code]

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

/*
Roll No: 20CS8016

Q4: Write a C program to perform matrix multiplication.
Use dynamic allocation 2-D arrays to store the matrices.
[Use malloc and free and make sure you explicitly clear the garbage after processing.]
*/

int **arr1;
int rows1, columns1;
int **arr2;
int rows2, columns2;


int ** multiply(int ** arr1, int ** arr2) {

  int ** result;
  result = (int **) malloc (sizeof(int *) * rows2);
  for (int i = 0; i < rows2; i++){
    result[i] = (int *) malloc (sizeof(int) * columns1);
  }

  //Result Matrix is of Order r1 * c2;
  for (int i = 0; i < rows1; i++)
    for (int j = 0; j < columns2; j++)
      result[i][j] = 0;

  for (int i = 0; i < rows1; i++){
    for (int j = 0; j < columns2; j++){
      for (int k = 0; k < columns1; k++){
        result[i][j] += arr1[i][k] * arr2[k][j];
      }
    }
  }

  return result;
}


int ** values_in (int ** arr, int rows, int columns){

  printf("Enter the values: ");

  arr = (int **) malloc (sizeof(int *) * rows);
  for (int i = 0; i < rows; i++){
    arr[i] = (int *) malloc (sizeof(int) * columns);
  }

  for (int i = 0; i < rows; i++){
    for (int j = 0; j < columns; j++){
      scanf("%d", &arr[i][j]);
    }
  }

  return arr;
}
```

```c


void display_matrix(int ** matrix, int rows, int columns) {

  printf("\n");

  for (int i = 0; i < rows; i++){
    printf("|");
    for (int j = 0; j < columns; j++){
      printf(" %d ", matrix[i][j]);
    }
    printf("|\n");
  }

  printf("\n");
}


void deallocate_memory(int ** result){

  for (int i = 0; i < rows1; i++)
    free(arr1[i]);
  free(arr1);
  for (int i = 0; i < rows2; i++)
    free(arr2[i]);
  free(arr2);
  for (int i = 0; i < rows1; i++)
    free(result[i]);
  free(result);

  return;
}


int main () {

  printf("\nEnter order of first matrix: ");
  scanf("%d %d", &rows1, &columns1);
  arr1 = values_in(arr1, rows1, columns1);

  printf("First Matrix is: \n");
  display_matrix(arr1, rows1, columns1);

  printf("Enter order of second matrix: ");
  scanf("%d %d", &rows2, &columns2);


  if (rows2 != columns1) {
    printf("Second Matrix Rows should be == First matrix's column: %d.\n", columns1);
    printf("Exiting Program...\n");
    return 0;
  } else {
    arr2 = values_in(arr2, rows2, columns2);
  }
  printf("Second Matrix is: \n");
  display_matrix(arr2, rows2, columns2);

  int ** result = multiply(arr1, arr2);
  printf("Result:\n");
  display_matrix(result, rows1, columns2);

  deallocate_memory(result);

  return 0;
}
```

<< Output Case 1

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments> cd "d:\Classes\Third Semester\DS
A Laboratory Assignments\Dynamic Allocation\" ; if ($?) { gcc matrix_multiplication.c -o
matrix_multiplication } ; if ($?) { .\matrix_multiplication }

Enter order of first matrix: 3 3
Enter the values: 1 6 4 2 3 9 7 8 5
First Matrix is:

| 1  6  4 |
| 2  3  9 |
| 7  8  5 |

Enter order of second matrix: 3 3
Enter the values: 1 0 0 0 1 0 0 0 1
Second Matrix is:

| 1  0  0 |
| 0  1  0 |
| 0  0  1 |

Result:

| 1  6  4 |
| 2  3  9 |
| 7  8  5 |
```

<< Output Case 2

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments> cd "d:\Classes\Third Semester\DS
A Laboratory Assignments\Dynamic Allocation\" ; if ($?) { gcc matrix_multiplication.c -o
matrix_multiplication } ; if ($?) { .\matrix_multiplication }

Enter order of first matrix: 2 3
Enter the values: 1 5 6 3 2 4
First Matrix is:

| 1  5  6 |
| 3  2  4 |

Enter order of second matrix: 1 3
Second Matrix Rows should be == First matrix's column: 3.
Exiting Program...
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Dynamic Allocation>
```

Answer 5

Determinant of a Matrix - [Code]

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

/*
Roll No: 20CS8016

Write a C program to find the determinant of a matrix.
Use dynamic allocation 2-D arrays to store the matrix.
Hint: Try to write a recursive function for computing determinant of a matrix.
*/

static int **arr;
int side;

int calculate_determinant(int ** arr, int side) {

  int determinant;
  int power = 1;
  if (side == 1){
    return (arr[0][0]);
  } else {
    determinant = 0;
    int ** minor; int size = side - 1;
    for (int axis = 0; axis < side; axis++){
      int row = 0, column = 0;

      minor = (int **) malloc (sizeof(int *) * (size));
      for (int i = 0; i < size; i++){
        minor[i] = (int *) malloc (sizeof(int) * (size));
      }

      for (int i = 0; i < side; i++){
        for (int j = 0; j < side; j++){
          if (i != 0 && j != axis){
            minor[row][column] = arr[i][j];

            if (column < side - 2){
              column++;
            } else {
              column = 0;
              row++;
            }
          }
        }
      }

      determinant += power * (arr[0][axis] * calculate_determinant(minor, size));
      power *= (-1);
    }
  }

  return determinant;
}

```

```c

void insert_values_of_matrix (){

  printf("Enter the values below: \n");

  arr = (int **) malloc (sizeof(int *) * side);
  for (int i = 0; i < side; i++){
    arr[i] = (int *) malloc (sizeof(int) * side);
  }

  //Input
  for (int i = 0; i < side; i++){
    for (int j = 0; j < side; j++){
      scanf("%d", &arr[i][j]);
    }
  }
}



int main () {

  printf("Enter order of matrix: ");
  scanf("%d", &side);

  insert_values_of_matrix();

  int determinant = calculate_determinant(arr, side);
  printf("Determinant: %d\n", determinant);

  return 0;
}
```

<< Output Case 1

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments> cd "d:\Classes\Third Semester\DS
A Laboratory Assignments\Submissions\Assignment 2\" ; if ($?) { gcc question5.c -o questi
on5 } ; if ($?) { .\question5 }
Enter order of matrix: 3
Enter the values below:
1 4 6
4 2 3
7 6 1
Determinant: 112
```

<< Output Case 2

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments> cd "d:\Classes\Third Semester\DS
A Laboratory Assignments\Submissions\Assignment 2\" ; if ($?) { gcc question5.c -o questi
on5 } ; if ($?) { .\question5 }
Enter order of matrix: 3
Enter the values below:
1 0 0
0 1 0
0 0 1
Determinant: 1
```

Answer 6

CGPA Database - [Code]

```c
#include <stdio.h>
#include <stdlib.h>
#include <windef.h>
#define departmentName database[i].name_of_dept
#define totalStudents database[i].cgpa[0]
#define grade database[i].cgpa
#define highestCGPA database[i].highest_gpa
#define highScorers database[i].highest_scorers
#define N 2


/*
Roll No: 20CS8016

Q6: Write a C program to store the CGPA obtained by students of different dept.
Note that Institute is having a fixed number of Departments
(i.e. the number of Depts are known prior to execution)
but the number of students in each Dept is known in runtime.
Moreover different Dept. have different student capacity.
Compute the highest CGPA obtained in each Dept. and highest CGPA among all Dept.

Hint: Using dynamic allocation, use malloc and free.
Make sure you explicitly clear the garbage after processing is done.
*/


typedef struct department {
  char name_of_dept[5];
  float * cgpa;
  float highest_gpa;
  int * highest_scorers;
} department;


department database[N];


void displayDatabase() {
  for (int i = 0; i < N; i++){
    printf("\n%s: \n\n", departmentName);
    printf("RollNo.  CGPA\n");
    for (int roll = 1; roll <= totalStudents; roll++){
      printf("     %d. %.3f\n", roll, grade[roll]);
    }
    printf("Highest CGPA: %.3f, RollNo(s): ", highestCGPA);
    for (int index = 1; index <= highScorers[0]; index++){
      printf("%d ", highScorers[index]);
    }
    printf("\n\n============\n");
  }
}


```

```c
void fillHighestGPA(){
  for (int i = 0; i < N; i++){

    for (int roll = 1; roll <= totalStudents; roll++){
      highestCGPA = max(highestCGPA, grade[roll]);
    }

    int no_of_student = 0;
    for (int roll = 1; roll <= totalStudents; roll++){
      if (grade[roll] == highestCGPA){
        no_of_student++;
      }
    }

    highScorers = (int *) malloc (sizeof(int) * (no_of_student + 1));
    highScorers[0] = no_of_student;

    for (int roll = 1, index = 1; roll <= totalStudents; roll++){
      if (grade[roll] == highestCGPA){
        highScorers[index] = roll;
        index++;
      } else {
        continue;
      }
    }
  }
}


void deallocateMemory() {

  for (int i = 0; i < N; i++){
    free(grade);
    free(highScorers);
  }
}


int main(){

  for (int i = 0; i < N; i++){
    printf("Department Name: ");
    scanf("%s", departmentName);

    printf("Number of Students: ");
    int size; scanf("%d", &size);

    grade = (float *)malloc (sizeof(float) * (size + 1));
    totalStudents = size;

    printf("Enter Grades below\n");
    for (int roll = 1; roll <= size; roll++){
      printf("     %d. ", roll);
      scanf("%f", &grade[roll]);
    }
  }

  fillHighestGPA();

  freopen("output.txt", "w", stdout);
  displayDatabase();
  deallocateMemory();

  return 0;
}
```

>> Input

```
PS D:\Classes\Third Semester\DSA Laboratory Assignments> cd "d:\Classes\Third Semester
\DSA Laboratory Assignments\Submissions\Assignment 2\" ; if ($?) { gcc question6.c -o
question6 } ; if ($?) { .\question6 }
Department Name: CSE
Number of Students: 5
Enter Grades below
     1. 5.69
     2. 3.94949494
     3. 6
     4. 9.88
     5. 9.88
Department Name: BT
Number of Students: 3
Enter Grades below
     1. 8.99
     2. 8.09
     3. 8.99
PS D:\Classes\Third Semester\DSA Laboratory Assignments\Submissions\Assignment 2>
```

<< Output

```
CSE:
RollNo.  CGPA
     1. 5.690
     2. 3.949
     3. 6.000
     4. 9.880
     5. 9.880

Highest CGPA: 9.880, RollNo(s): 4 5

=============
BT:
RollNo.  CGPA
     1. 8.990
     2. 8.090
     3. 8.990

Highest CGPA: 8.990, RollNo(s): 1 3

=============
```