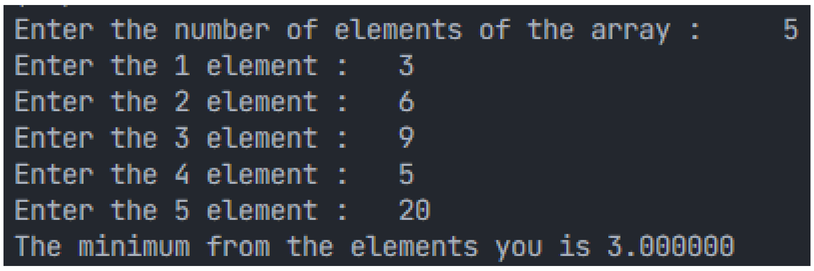
DS: PRACTICAL 2

NAME: Vivian Viajy Ludrick
ROLL NO. : 9914
BRANCH : S.E. Comps-A
BATCH : C

Q.1. WAP to create an array of float of size n, and find minimum in an array.

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

// checks whether there is space available or not
void checkSpace(float *arr)
{
  if (arr == NULL)
  {
    printf("Error: Insufficient memory. Cannot proceed with the
execution\n");
    exit(1);
  }
}

// enters the elements in an array
void setElements(float *arr, int n)
{
  for (int i = 0; i < n; i++)
  {
    printf("Enter the %d element :\t", i + 1);
    scanf("%f", &arr[i]);
  }
}

// return the minimum value in the given array
float setMin(float *arr, int n)
{
  float min = arr[0];
  for (int i = 1; i < n; i++)
  {
    if (arr[i] < min)
    {
      min = arr[i];
    }
  }
  return min;
}

int main()
{

  int n; // number of elements of the array
  printf("Enter the number of elements of the array :\t");
  scanf("%d", &n);
  float *arr = (float *)malloc(n * sizeof(float)); // allocating a
dynamically created array to arr[n]
  checkSpace(arr);
  setElements(arr, n);

  float minimum = setMin(arr, n);
  printf("The minimum from the elements you is %f", minimum);

  free(arr); // frees the allocated memory
  return 0;
}
```

OUTPUT :



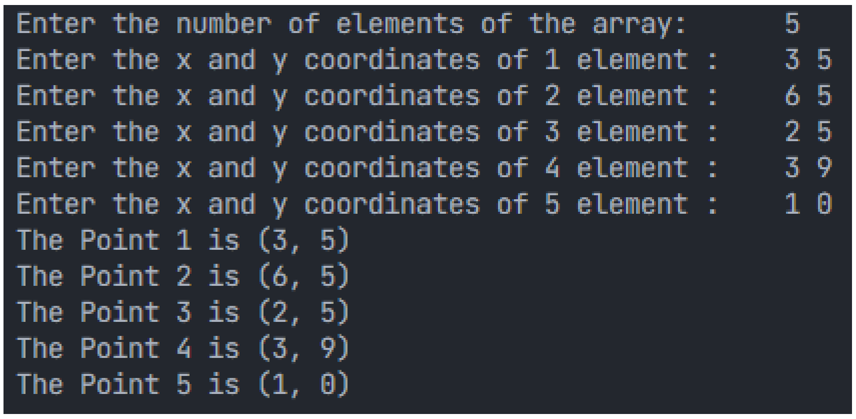
```
Enter the number of elements of the array :     5
Enter the 1 element :    3
Enter the 2 element :    6
Enter the 3 element :    9
Enter the 4 element :    5
Enter the 5 element :    20
The minimum from the elements you is 3.000000
```

Q.2. WAP to create array of n points.

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

// Point structure
typedef struct
{
  int x;
  int y;
} Point;

// checks whether there is space available or not
void checkSpace(Point *points)
{
  if (points == NULL)
  {
    printf("Error!!!: Insufficient memory. Cannot proceed with the
execution\n");
    exit(1);
  }
}

// sets the elements of the array
void setElements(Point *points, int n)
{
  for (int i = 0; i < n; i++)
  {
    printf("Enter the x and y coordinates of %d element :\t", i +
1);
    // &(points[i].x) == &(*(points + i)).x == &((points + i)->x)
    // x->y indicates *x.y since y is linked to the value of x and
not the address of x
    // x[i].y indicates (*(x + i)).y (this is a value) increment
address x by i(size of datatype is automatically determined) then get
its value and then get the x element linkedto its value which returns
a value.
```

```
    scanf("%d%d", &(points[i].x), &(points[i].y));
  }
}

// prints the elements of the array
void getElements(Point *points, int n)
{
  for (int i = 0; i < n; i++)
  {
    printf("The Point %d is (%d, %d)\n", i + 1, points[i].x,
(points[i].y));
  }
}
int main()
{
  int n; // number of elements of the array
  printf("Enter the number of elements of the array:\t");
  scanf("%d", &n);

  // (datatype *)calloc(number of elements, size of datatype)
  Point *points = (Point *)calloc(n, sizeof(Point)); // allocating a
dynamically created array to arr[n]

  checkSpace(points);

  setElements(points, n);
  getElements(points, n);

  free(points);
  return 0;
}
```

OUTPUT :



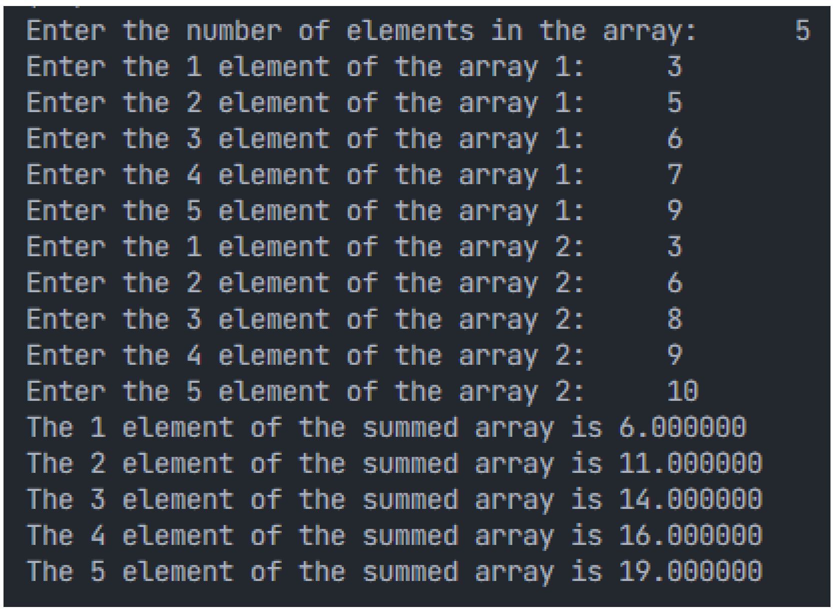
```
Enter the number of elements of the array:       5
Enter the x and y coordinates of 1 element :    3 5
Enter the x and y coordinates of 2 element :    6 5
Enter the x and y coordinates of 3 element :    2 5
Enter the x and y coordinates of 4 element :    3 9
Enter the x and y coordinates of 5 element :    1 0
The Point 1 is (3, 5)
The Point 2 is (6, 5)
The Point 3 is (2, 5)
The Point 4 is (3, 9)
The Point 5 is (1, 0)
```

Q.3. WAP to create add two array, create it dynamically

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

// checks whether there is memory available or not for the dynamic array
int checkMemory(float *a)
{
  if (a == NULL)
  {
    printf("Error!!!: Insufficient memory. Cannot proceed with the execution\n");
    exit(1);
  }
}

// sets the elements of the dynamic array
void setElements(float *a, int n, int arrNum)
{
  for (int i = 0; i < n; i++)
  {
    printf("Enter the %d element of the array %d:\t", i + 1, arrNum);
    scanf("%f", &a[i]);
  }
}
```

```c
/*
definition: adding two arrays and storing it in another array
useCase: addingBothArrays(float *sum, float *a, float *b, int n)
---------------------------
here sum[i] = a[i] + b[i] occurs n times
*/
void addingBothArrays(float *sum, float *a, float *b, int n)
{
  for (int i = 0; i < n; i++)
  {
    sum[i] = a[i] + b[i];
  }
}

// prints the elements of the dynamic array
void getElements(float *a, int n)
{
  for (int i = 0; i < n; i++)
  {
    printf("The %d element of the summed array is %f\n", i + 1,
a[i]);
  }
}
  int main()
  {
    int n; // number of elements in the arrays
    printf("Enter the number of elements in the array:\t");
    scanf("%d", &n);
    float *arr1 = (float *)malloc(n * sizeof(float));   // dynamic
array 1
    float *arr2 = (float *)calloc(n, sizeof(float));    // dynamic
arr 2
    float *sumArr = (float *)malloc(n * sizeof(float)); // dynamic
sum of arr1 and arr2
    checkMemory(arr1);
    checkMemory(arr2);

    setElements(arr1, n, 1);
    setElements(arr2, n, 1);

    addingBothArrays(sumArr, arr1, arr2, n);

    getElements(sumArr,n);

    free(arr1);
    free(arr2);
    free(sumArr);
    return 0;
}
```

OUTPUT :



```
Enter the number of elements in the array:      5
Enter the 1 element of the array 1:      3
Enter the 2 element of the array 1:      5
Enter the 3 element of the array 1:      6
Enter the 4 element of the array 1:      7
Enter the 5 element of the array 1:      9
Enter the 1 element of the array 2:      3
Enter the 2 element of the array 2:      6
Enter the 3 element of the array 2:      8
Enter the 4 element of the array 2:      9
Enter the 5 element of the array 2:      10
The 1 element of the summed array is 6.000000
The 2 element of the summed array is 11.000000
The 3 element of the summed array is 14.000000
The 4 element of the summed array is 16.000000
The 5 element of the summed array is 19.000000
```

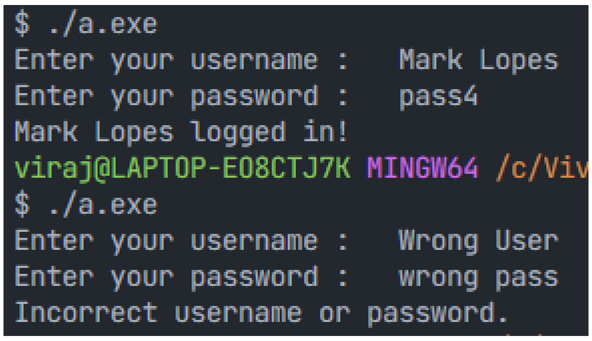
Q.4. Create a database of n users having username and password, search whether a new user entering in google meet, is authorized or not.

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

#define USERNAME_SIZE 20 // size of user name is 20 characters
#define PASSWORD_SIZE 50 // size of user password is 50 characters
#define NO_OF_USERS 5    // number of users in the database

// structure to store the user info efficiently
typedef struct
{
  char username[USERNAME_SIZE]; // the name with which the user will
be identified
  char password[PASSWORD_SIZE]; // used for user account verification
} User;

// checks whether there is space available or not
void checkMemory(User *curUser)
{
  if (curUser == NULL)
  {
    printf("Error!!!: Insufficient memory. Cannot proceed with the
execution\n");
    exit(1);
  }
}

// sets the username and password for the user trying to login
void setCurUser(User *curUser)
{
  printf("Enter your username :\t");
  // always use a space before inserting in a string so that it
consumes the new line character. Also dont put space at the end of
the string as it will infinitely take input.
```

```c
  scanf(" %[^\n]", curUser->username);
  //can't use fgets as there is some issue with the newline character
that gets inserted so using the scanf workaround instead
  printf("Enter your password :\t");
  scanf(" %[^\n]", curUser->password);
}

/* checks whether the user trying to login
  is allowed to enter the room or not.
  ---------------------------------------
  match found ? return 0 : return 1

*/
int checkUserValidity(User curUser, User *userdb)
{
  for (int i = 0; i < NO_OF_USERS; i++)
  {
    if (strcmp(curUser.username, userdb[i].username) == 0 &&
strcmp(curUser.password, userdb[i].password) == 0)
    {
      return 0;
    }
  }
  return 1;
}

int main()
{
  // deliberately not initialized dynamically since i wanted to test
this syntax
  User userdb[NO_OF_USERS] = {{"Pratyay Koley", "pass1"}, {"Shaun
Mendes", "pass2"}, {"Shwen Coutinho", "pass3"}, {"Mark Lopes",
"pass4"}, {"Jonathan Gomes", "pass5"}};

  User *curUser = (User *)malloc(sizeof(User)); // user trying to
login in the meeting room
```

```c
  checkMemory(curUser);

  setCurUser(curUser);

  int flagValidity = checkUserValidity(*curUser, userdb);

  flagValidity == 0 ? printf("%s logged in!", curUser->username) : printf("Incorrect username or password.");

  free(curUser);
  return 0;
}
```

OUTPUT :

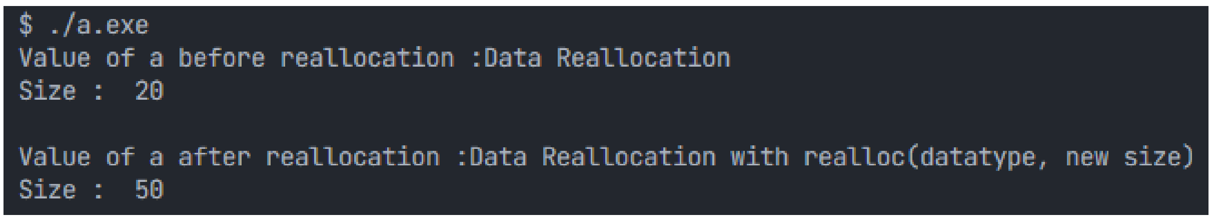


```
$ ./a.exe
Enter your username :   Mark Lopes
Enter your password :   pass4
Mark Lopes logged in!
viraj@LAPTOP-EO8CTJ7K MINGW64 /c/Viv
$ ./a.exe
Enter your username :   Wrong User
Enter your password :   wrong pass
Incorrect username or password.
```

Q.5. WAP to program to show use of realloc function.

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

#define OLD_SIZE 20
#define NEW_SIZE 50

// checks whether there is space available or not
void checkMemory(char *ptr)
{
  if (ptr == NULL)
  {
    printf("Error!!!: Insufficient memory. Cannot proceed with the execution\n");
    exit(1);
  }
}

int main()
{
  char *a = (char *)malloc(OLD_SIZE * sizeof(char)); // assigned 'a' a location dynamically
  checkMemory(a);
  strcpy(a, "Data Reallocation"); // puts the right string in the left pointer as as cannot assign a string the normal way
  printf("Value of a before reallocation :%s\nSize :\t%d\n\n", a, OLD_SIZE);

  a = (char *)realloc(a, NEW_SIZE * sizeof(char)); // reallocated a new size memory to 'a'
  checkMemory(a);
  strcat(a, " with realloc(datatype, new size)"); //
  printf("Value of a after reallocation :%s\nSize :\t%d\n", a, NEW_SIZE);

  free(a); // free memory
  return 0;
}
```

OUTPUT :



```
$ ./a.exe
Value of a before reallocation :Data Reallocation
Size :   20

Value of a after reallocation :Data Reallocation with realloc(datatype, new size)
Size :   50
```