Name - Chiradeep Banik Enroll - 20UCS176

Sec - A

Q.1 Create a structure for a company who manufactures mobile phones. Create a repository of information which consists of the following data:

a. prod_name
b. prod_code
c. prod_price
d. prod_screensize
e. prod_colore

Now, accept data for 5 different products and display them in a well formatted manner. Use structure and arrays.

→

```
#include <stdio.h>

struct company_repod
    char prod_name[20];
    int prod_code;
    float prod_price;
    int prod_screensize;
    char prod_colore[20];
};
```

```
void take_input(struct company_repo *repo){
    for(int i=0; i<5; i++){
        printf("\nEnter the product name: ");
        scanf("%s", &(repo+i)->prod_name);
        printf("Enter the product code : ");
        scanf("%d", &(repo+i)->prod_code);
        printf("Enter the product price : ");
        scanf("%f", &(repo+i)->prod_price);
        printf("Enter the product screensize : ");
        scanf("%d", &(repo+i)->prod_screensize);
        printf("Enter the product colon : ");
        scanf("%s", &(repo+i)->prod_colon);
    }
}
void display_output(struct company_repo *repo){
    for(int i=0; i<5; i++){
        printf("\nProduct name : %s", (repo+i)->prod_name);
        printf("\nProduct code : %d", (repo+i)->prod_code);
        printf("\nProduct price : %f", (repo+i)->prod_price);
        printf("\nProduct screensize : %d", (repo+i)->
                                    prod_screensize);
        printf("\nProduct colon : %s\n", (repo+i)->
                                    prod_colon);
    }
}
```

```
void main(){
    struct company_repo c_repo[5]
    take_input(c_repo);
    display_output(c_repo);
}
```

Console →

→ Enter the product name : P1
→ Enter the product code : 1
→ Enter the product price : 10
→ Enter the product screensize : 101
→ Enter the product color : red
→
→ Enter the product name : P2
→ Enter the product code : 2
→ Enter the product price : 20
→ Enter the product screensize : 202
→ Enter the product color : green

→

→
→ Enter the productname : P3
→ Enter the product code : 3
→ Enter the product price : 30
→ Enter the product screensite : 303
→ Enter the product colon : blue
→
→ Enter the product name : P4
→ Enter the product code : 4
→ Enter the product price : 40
→ Enter the product screensize : 404
→ Enter the colon : yellow
→
→ Enter the product name : P5
→ Enter the product code : 5
→ Enter the product price : 50
→ Enter the product screensize : 505
→ Enter the colon : black.

→ Product name : p1
Product code : 1
product price : 10.00000
Product screensize : 101
Product colon : red

Product name : p2
Product code : 2
Product price : 20.00000
Product screensize : 202
Product colon : green

Product name : p3
Product code : 3
Product price : 30.00000
Product screensize : 303
product colon : blue

Product name : p4
Product code : 4
Product price : 40.00000
Product screensize : 404
Product colon : yellow

Product name : p5

Product code : 5

Product price : 50.00000

Product screensize : 505

Product colour : black.

Q.2 Implement DMA to demonstrate the usage and functionalities of the following:

a. malloc()

b. calloc()

c. free()

→

```
#include <stdio.h>
#include <stdlib.h>
void main(){
    int *ptr = (int*) malloc(sizeof(int));
    // Do not set any default value.
    printf("After malloc -- %d\n", *ptr);
    int* ptr2 = (int*) calloc(1, sizeof(int));
    printf("After calloc -- %d\n", *ptr2);
    // sets the value as 0.
    free(ptr); // deallocates the memory.
    free(ptr2);
}
```

Console →

→ After malloc -- -78681314

→ After calloc -- 0

Q.3 Implement DMA to accept N integer data and find the sum of the composite integers only. Also display the memory allocations along with the data.

→
```
#include <stdio.h>
#include <stdlib.h>
void main(){
    int N;
    printf("Enter the value of N: ");
    scanf("%d", &N);
    int *arr = (int*) calloc(N, sizeof(int));
    for (int i=0; i<N; i++){
        printf("Enter the value of arr[%d]:", i);
        scanf("%d", (arr+i));
    }
```

→

```
int sum=0
for (int i=0 ; i<N ; i++) {
    printf("Memory location of arr[%d] — %x \n",
                                    i , (arr+i));
    int c=0
    for (int j=1 ; j<=*(arr+i) ; j++) {
        if (*(arr+i) % j == 0) {
            c++;
        }
    }
    if (c>2) {
        sum += *(arr+i);
    }
}
printf("Sum of the composite integers in
        the array is %d\n", sum);
}
```

Console →

→ Enter the value of N : 5

→ Enter the value of arr[0] : 1

→ Enter the value of arr[1] : 2

→ Enter the value of arr[2] : 3

→ Enter the value of arr[3] : 9

→

→ Enter the value of arr[4] : 5

→ Memory location of arr[0] -- 1058ca0

Memory location of arr[1] -- 1058ca4

Memory location of arr[2] -- 1058ca8

Memory location of arr[3] -- 1058cac

Memory location of arr[4] -- 1058cb0

Sum of the composite integers in the array is 9

Q.9 Implement DMA and create a memory location of size N. Accept the data by passing the base address of the allocation to a function and display them along with the memory locations inside the function.

→
```
#include <stdio.h>
#include <stdlib.h>
void take_input (int *ptr, int n){
    for(int i=0 ; i<n ; i++){
        printf("Enter the value at %d :", i);
        scanf("%d", (ptr+i));
    }
}
```
→

```
void show_input(int* ptr, int n){
    for(int j=0; j<n; j++){
        printf("Index - %d Memory Address -- %p
                value - %d\n", j, (ptr+j), *(ptr+j));
    }
}
void main(){
    int N;
    printf("Enter the unit size of the memory
                to be allocated : ");
    scanf("%d", &N);
    int *ptr = (int*) malloc(N* sizeof(int));
    take_input(ptr, N);
    show_input(ptr, N);
}
```

Console →

→ Enter the unit size of the memory to be allocated : 3
→ Enter the value at 0 : 1
→ Enter the value at 1 : 2
→ Enter the value at 2 : 3
→ Index - 0 - Memory Address -- 0x55b18edb9ac0 value --
Index - 1 - Memory Address -- 0x55b18edb9ac4 value - 2
Index - 2 - Memory Address -- 0x55b18edb9ac8 value - 3