P1→

Design, Develop and Implement a menu driven Program in C for the following array operations.

a) Creating an array of N Integer Elements

b) Display of array Elements with Suitable Headings

c) Inserting an Element (ELEM) at a given valid Position (POS)

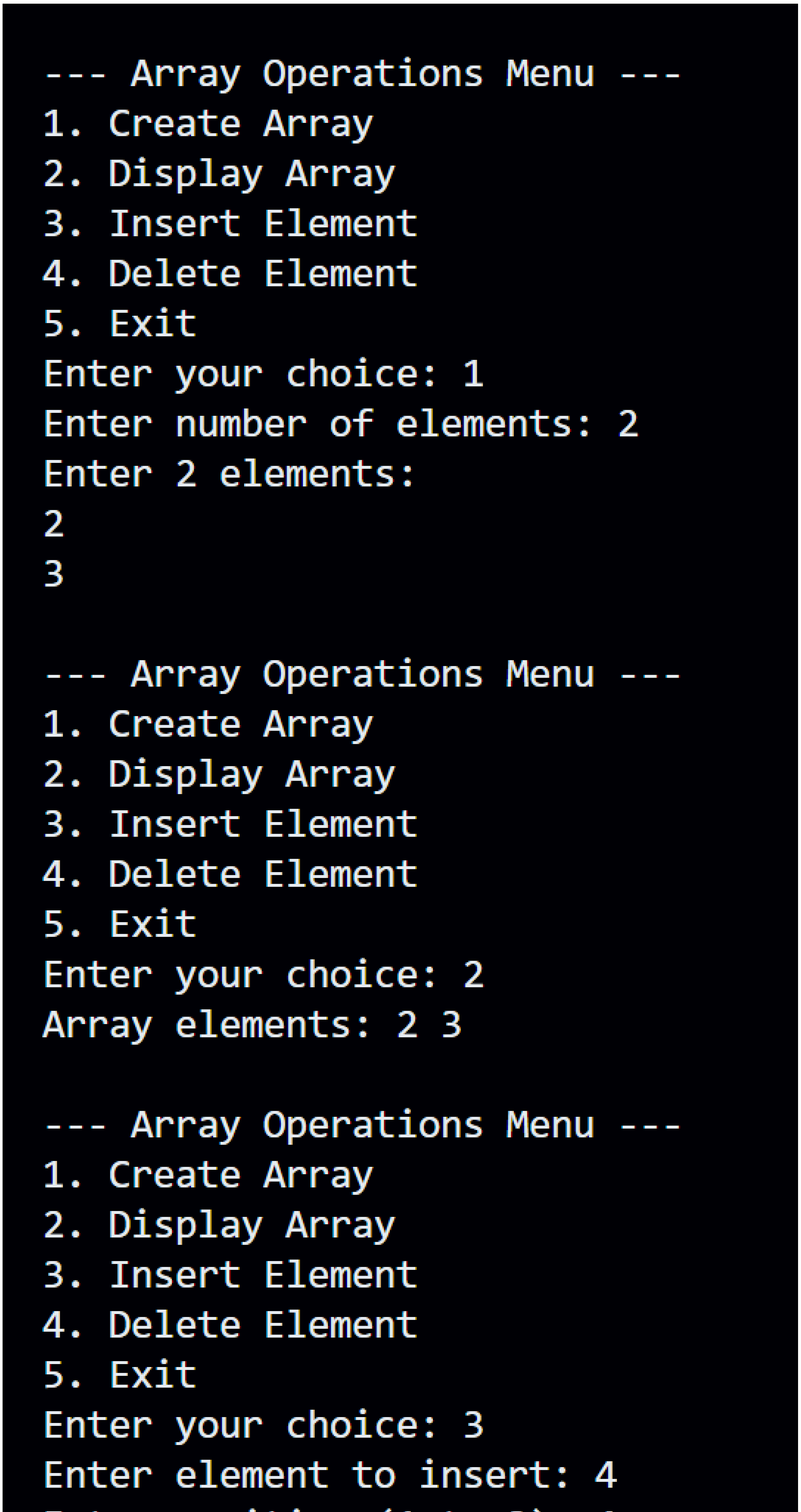
d) Deleting an Element at a given valid Position (POS)

e) Exit.

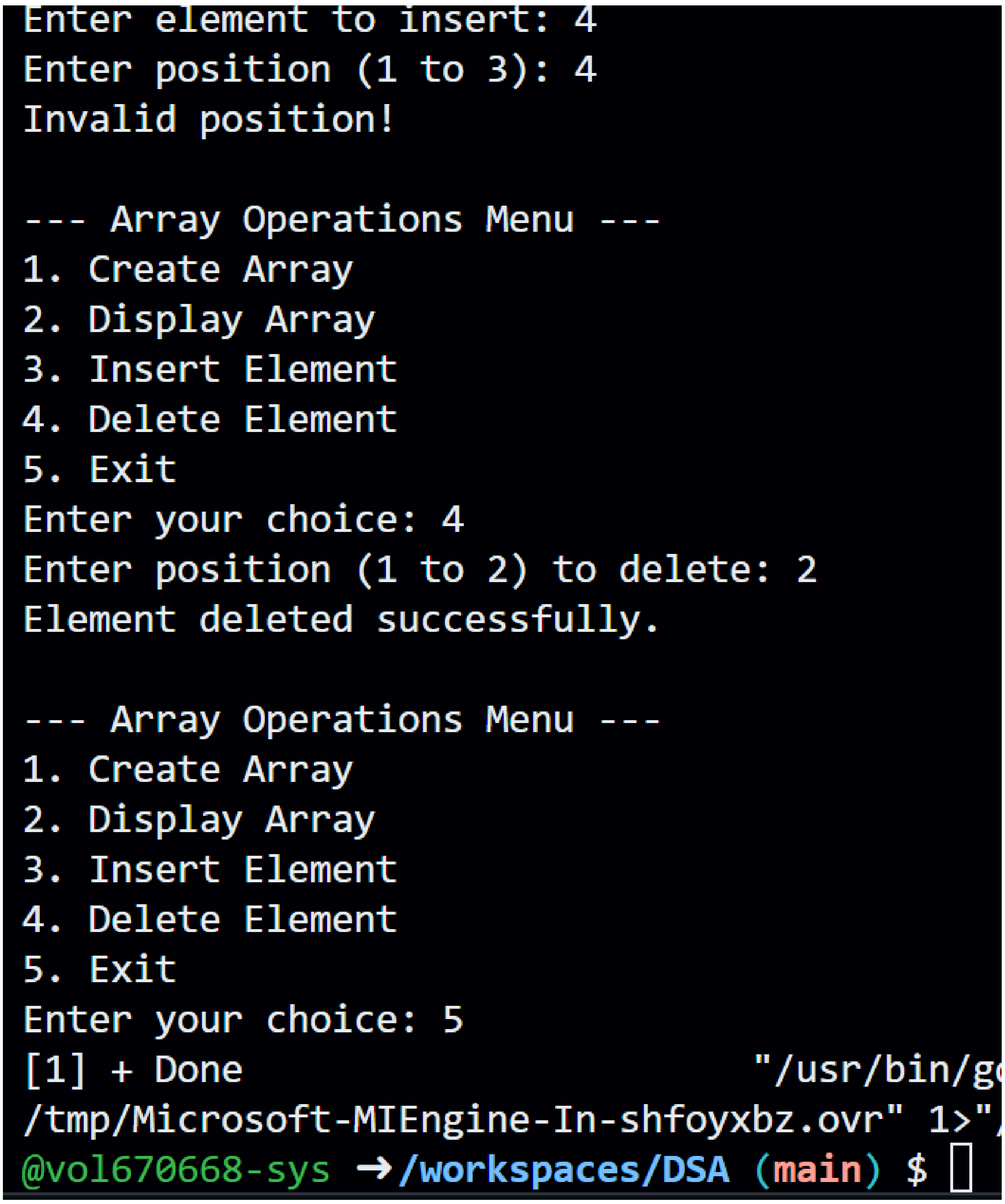
Support the program with functions for each of the above operations.

code—>>>>

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

#define MAX 100  // Maximum array size

// Function prototypes
void createArray(int arr[], int *n);
void displayArray(int arr[], int n);
void insertElement(int arr[], int *n, int elem, int pos);
void deleteElement(int arr[], int *n, int pos);

int main() {
   int arr[MAX];
   int n = 0, choice, elem, pos;

   while (1) {
      printf("\n--- Array Operations Menu ---\n");
      printf("1. Create Array\n");
      printf("2. Display Array\n");
      printf("3. Insert Element\n");
      printf("4. Delete Element\n");
      printf("5. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            createArray(arr, &n);
            break;
         case 2:
            displayArray(arr, n);
            break;
         case 3:
            printf("Enter element to insert: ");
            scanf("%d", &elem);
            printf("Enter position (1 to %d): ", n + 1);
            scanf("%d", &pos);
            insertElement(arr, &n, elem, pos);
            break;
         case 4:
            printf("Enter position (1 to %d) to delete: ", n);
            scanf("%d", &pos);
            deleteElement(arr, &n, pos);
            break;
         case 5:
            exit(0);
         default:
            printf("Invalid choice! Try again.\n");
      }
   }
   return 0;
}

// Function to create array
void createArray(int arr[], int *n) {
   int i;
   printf("Enter number of elements: ");
   scanf("%d", n);

   if (*n > MAX) {
      printf("Maximum size allowed is %d\n", MAX);
      *n = 0;
      return;
   }

   printf("Enter %d elements:\n", *n);
   for (i = 0; i < *n; i++) {
      scanf("%d", &arr[i]);
   }
}

// Function to display array
void displayArray(int arr[], int n) {
   int i;
   if (n == 0) {
      printf("Array is empty.\n");
      return;
   }
   printf("Array elements: ");
   for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

// Function to insert an element
void insertElement(int arr[], int *n, int elem, int pos) {
    int i;
    if (*n == MAX) {
        printf("Array is full, cannot insert.\n");
        return;
    }
    if (pos < 1 || pos > *n + 1) {
        printf("Invalid position!\n");
        return;
    }

    for (i = *n; i >= pos; i--) {
        arr[i] = arr[i - 1];
    }
    arr[pos - 1] = elem;
    (*n)++;
    printf("Element inserted successfully.\n");
}

// Function to delete an element
void deleteElement(int arr[], int *n, int pos) {
    int i;
    if (*n == 0) {
        printf("Array is empty, nothing to delete.\n");
        return;
    }
    if (pos < 1 || pos > *n) {
        printf("Invalid position!\n");
        return;
    }

    for (i = pos - 1; i < *n - 1; i++) {
        arr[i] = arr[i + 1];
    }
    (*n)--;
    printf("Element deleted successfully.\n");
}
```



```
--- Array Operations Menu ---
1. Create Array
2. Display Array
3. Insert Element
4. Delete Element
5. Exit
Enter your choice: 1
Enter number of elements: 2
Enter 2 elements:
2
3

--- Array Operations Menu ---
1. Create Array
2. Display Array
3. Insert Element
4. Delete Element
5. Exit
Enter your choice: 2
Array elements: 2 3

--- Array Operations Menu ---
1. Create Array
2. Display Array
3. Insert Element
4. Delete Element
5. Exit
Enter your choice: 3
Enter element to insert: 4
```



```
Enter element to insert: 4
Enter position (1 to 3): 4
Invalid position!

--- Array Operations Menu ---
1. Create Array
2. Display Array
3. Insert Element
4. Delete Element
5. Exit
Enter your choice: 4
Enter position (1 to 2) to delete: 2
Element deleted successfully.

--- Array Operations Menu ---
1. Create Array
2. Display Array
3. Insert Element
4. Delete Element
5. Exit
Enter your choice: 5
[1] + Done                      "/usr/bin/g
/tmp/Microsoft-MIEngine-In-shfoyxbz.ovr" 1>"
@vol670668-sys →/workspaces/DSA (main) $ 
```

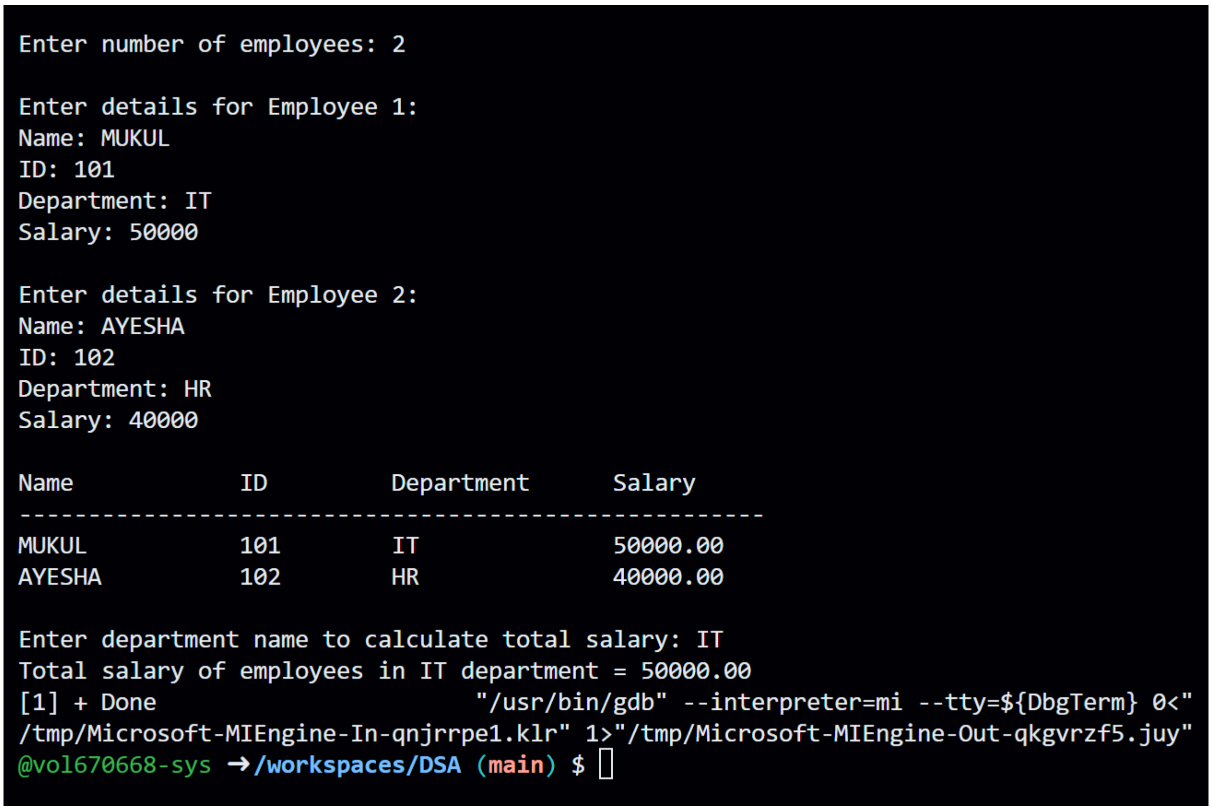
P2—>

Define an EMPLOYEE structure with members Emp_name, Emp-id, Dept-name and Salary. Read and display data of N employees. Employees may belong to different departments. Write a function to find total salary of employees of a specified department. Use the concept of pointer to structure and allocate the memory dynamically to EMPLOYEE instances

CODE=>

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

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

// Structure definition
struct EMPLOYEE {
    char Emp_name[50];
    int Emp_id;
    char Dept_name[50];
    float Salary;
};

// Function prototypes
void readEmployees(struct EMPLOYEE *e, int n);
void displayEmployees(struct EMPLOYEE *e, int n);
float totalSalaryByDept(struct EMPLOYEE *e, int n, char dept[]);

int main() {
    struct EMPLOYEE *emp;
    int n, i;
    char dept[50];

    printf("Enter number of employees: ");
    scanf("%d", &n);

    // Dynamic allocation for n employees
    emp = (struct EMPLOYEE *)malloc(n * sizeof(struct EMPLOYEE));
    if (emp == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    // Read and display employee details
    readEmployees(emp, n);
    displayEmployees(emp, n);

    // Salary computation for a department
    printf("\nEnter department name to calculate total salary: ");
    scanf("%s", dept);

    float total = totalSalaryByDept(emp, n, dept);
    printf("Total salary of employees in %s department = %.2f\n", dept, total);

    free(emp); // Free allocated memory
    return 0;
}

// Function to read employee data
void readEmployees(struct EMPLOYEE *e, int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("\nEnter details for Employee %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", (e + i)->Emp_name);
        printf("ID: ");
        scanf("%d", &(e + i)->Emp_id);
        printf("Department: ");
        scanf("%s", (e + i)->Dept_name);
        printf("Salary: ");
        scanf("%f", &(e + i)->Salary);
    }
}

// Function to display employee data
void displayEmployees(struct EMPLOYEE *e, int n) {
    int i;
    printf("\n%-15s %-10s %-15s %-10s\n", "Name", "ID", "Department", "Salary");
    printf("------------------------------------------------------\n");
    for (i = 0; i < n; i++) {
        printf("%-15s %-10d %-15s %-10.2f\n",
               (e + i)->Emp_name,
               (e + i)->Emp_id,
               (e + i)->Dept_name,
               (e + i)->Salary);
    }
}

// Function to calculate total salary by department
float totalSalaryByDept(struct EMPLOYEE *e, int n, char dept[]) {
    float sum = 0.0;
    int i;
    for (i = 0; i < n; i++) {
        if (strcmp((e + i)->Dept_name, dept) == 0) {
            sum += (e + i)->Salary;
        }
    }
    return sum;
}
```



```
Enter number of employees: 2

Enter details for Employee 1:
Name: MUKUL
ID: 101
Department: IT
Salary: 50000

Enter details for Employee 2:
Name: AYESHA
ID: 102
Department: HR
Salary: 40000

Name            ID          Department      Salary
-------------------------------------------------------
MUKUL           101         IT              50000.00
AYESHA          102         HR              40000.00

Enter department name to calculate total salary: IT
Total salary of employees in IT department = 50000.00
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"
/tmp/Microsoft-MIEngine-In-qnjrrpe1.klr" 1>"/tmp/Microsoft-MIEngine-Out-qkgvrzf5.juy"
@vol670668-sys ➜ /workspaces/DSA (main) $ 
```