Batch: C1_2	**Roll No.: 16010123032**
Experiment / assignment / tutorial No. 7	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Write a program in C to demonstrate use of structures and union.

AIM: Write a program to manage an employee database using structure and union in C. Each Employee has the following information:

1. Employee ID(integer)
2. Name(string)
3. Department(string)
4. Salary(float)

You need to implement the following functionalities:

1. Create a structure named Employee with the appropriate data members to store the information mentioned above.
2. Create a union named EmployeeInfo that can hold either the Name or Department information.
3. Write a function addEmployee that takes user input for each employee's information and stores it in an array of structures.
4. Write a function printEmployeeDetails that takes an employee's ID as input and prints all available details for that employee.
5. Write a function updateEmployeeInfo that takes an employee's ID and allows the user to update either the Name or Department information using the EmployeeInfo union.
6. Implement a menu-driven program that allows the user to perform the above operations. Include options to add a new employee, print employee details, update employee information, and exit the program.

Expected OUTCOME of Experiment:

Design modular programs using functions and the use of structure and union(CO4).

Books/ Journals/ Websites referred:

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.

Problem Definition:
The program accepts a choice from the user using a switch case statement and generates output accordingly.

Algorithm:

1)Introduction:

This program manages employee records, allowing users to add new employees, view existing employee details, and update employee information.

2)Employee Structure:

Employees are represented by a structure containing their ID, name, department, and salary.

3)Functionality:

Adding Employees:Users can add new employees by providing their ID, name, department, and salary.

Printing Employee Details: Users can view the details of all employees currently in the system.

4)Updating Employee Information:

Users can update the name or department of an existing employee by searching for them using their ID.

5)Main Program Flow:

I)Initialization:

Users are prompted to specify the number of employees they want to manage.

II)User Interaction Loop:

The program enters a loop where users are presented with options:

Add Employee: Allows users to input details for new employees.

Print Employee Details: Displays the details of all employees.

Update Employee Info: Enables users to modify the name or department of an existing employee.

Exit: Terminates the program when selected.

Exiting the Program:

The program concludes when the user chooses to exit, providing a clean end to the interaction.

6)User Experience:

Users interact with the program by selecting options based on the actions they want to perform.

The program guides users through each step, ensuring clarity in input and output.

7)Termination:

The program gracefully exits upon the user's request, allowing them to manage employee records efficiently.

Implementation details:

```
#include <stdio.h>
#include <stdlib.h>
struct employee
{
  int id;
  char name[20];
  char dept[20];
  float salary;
};
void add_employee(struct employee emp[], int n)
{
  for (int i = 0; i < n; i++)
  {
    printf("Enter employee id: ");
    scanf("%d", &emp[i].id);
    printf("Enter the employee name: ");
    scanf("%s", emp[i].name);
    printf("Enter the employee department: ");
    scanf("%s", emp[i].dept);
    printf("Enter the employee salary: ");
    scanf("%f", &emp[i].salary);
  }
}
void print_employees(struct employee emp[], int n)
{
  for (int i = 0; i < n; i++)
  {
    printf("Employee %d\n", i + 1);
    printf("ID: %d\n", emp[i].id);
    printf("Name: %s\n", emp[i].name);
    printf("Department: %s\n", emp[i].dept);
    printf("Salary: %.2f\n", emp[i].salary);
```

```
    printf("\n");
  }
}
void update_employee(struct employee emp[], int n)
{
  int t, choice;
  printf("Enter the employee id you want to update: ");
  scanf("%d", &t);
  int employee_found = 0;
  for (int i = 0; i < n; i++)
  {
    if (t == emp[i].id)
    {
      employee_found = 1;
      printf("Employee found! Please select what you want to update:\n");
      printf("1. Name\n");
      printf("2. Department\n");
      scanf("%d", &choice);
      switch (choice)
      {
      case 1:
        printf("Enter new name: ");
        scanf("%s", emp[i].name);
        break;
      case 2:
        printf("Enter new department: ");
        scanf("%s", emp[i].dept);
        break;
      default:
        printf("Invalid choice\n");
      }
    }
  }
  if (!employee_found)
  {
    printf("Employee ID not found.\n");
  }
}
int main()
{
  int n;
  printf("Enter number of employees: ");
  scanf("%d", &n);
```

```
  struct employee emp[n];
  int choice;

  do
  {
    printf("Enter 1) Add employee, 2) Print employee details, 3) Update employee
info, or 4) Exit: ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      add_employee(emp, n);
      break;
    case 2:
      print_employees(emp, n);
      break;
    case 3:
      update_employee(emp, n);
      break;
    case 4:
      printf("Exiting the program.\n");
      break;
    default:
      printf("Invalid choice\n");
    }
  } while (choice != 4);
  return 0;
}
```

Output(s):

```
Aksh Maheshwari 16010123032
Enter number of employees: 2
Enter 1) Add employee, 2) Print employee details, 3) Update employee info, or 4) Exit: 1
Enter employee id: 32
Enter the employee name: Aksh
Enter the employee department: Mech
Enter the employee salary: 60000
Enter employee id: 4
Enter the employee name: Aakanksha
Enter the employee department: Comps
Enter the employee salary: 90000
Enter 1) Add employee, 2) Print employee details, 3) Update employee info, or 4) Exit: 2
Employee 1
ID: 32
Name: Aksh
Department: Mech
Salary: 60000.00

Employee 2
ID: 4
Name: Aakanksha
Department: Comps
Salary: 90000.00

Enter 1) Add employee, 2) Print employee details, 3) Update employee info, or 4) Exit: 3
Enter the employee id you want to update: 32
Employee found! Please select what you want to update:
1. Name
2. Department
2
Enter new department: Comps
Enter 1) Add employee, 2) Print employee details, 3) Update employee info, or 4) Exit: 2
Employee 1
ID: 32
Name: Aksh
Department: Comps
Salary: 60000.00

Employee 2
ID: 4
Name: Aakanksha
Department: Comps
Salary: 90000.00

Enter 1) Add employee, 2) Print employee details, 3) Update employee info, or 4) Exit: 4
Exiting the program.

Process returned 0 (0x0)   execution time : 64.297 s
Press any key to continue.
```

Conclusion:
We learnt about structures and unions in C and made a menu driven program that adds,prints,updates the employee details we stored in the structure.

Post Lab Descriptive Questions

- WAP to accept student name, roll number and percentage for 10 students using array of structures and arrange them in descending order of their percentage.

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

struct student_details
{
 int roll_no;
 char name[20];
 float marks;
};
void descending_sort(struct student_details details[], int n) {
   struct student_details temp;
   for (int i = 0; i < n; i++) {
      for (int j = i + 1; j < n; j++) {
         if (details[i].marks < details[j].marks) {
            temp = details[i];
            details[i] = details[j];
            details[j] = temp;
         }
```

```
        }
    }
}

int main()
{
  struct student_details details[10];
  for(int i=0;i <10;i++)
    {
      printf("Enter details for student %d\n",i+1);
      printf("Enter roll no: ");
      scanf("%d",&details[i].roll_no);
      printf("Enter name of student:");
      scanf("%s",details[i].name);
      printf("Enter marks for student in percentage out of 100: ");
      scanf("%f",&details[i].marks);

    }
  descending_sort(details,10);
  printf("Details of students in decreasing order of percentage:\n");
  for(int i=0;i<10;i++)
    {
         printf("Roll No: %d, Name: %s, Percentage: %.2f\n", details[i].roll_no,
details[i].name, details[i].marks);

    }
  return 0;

}
```

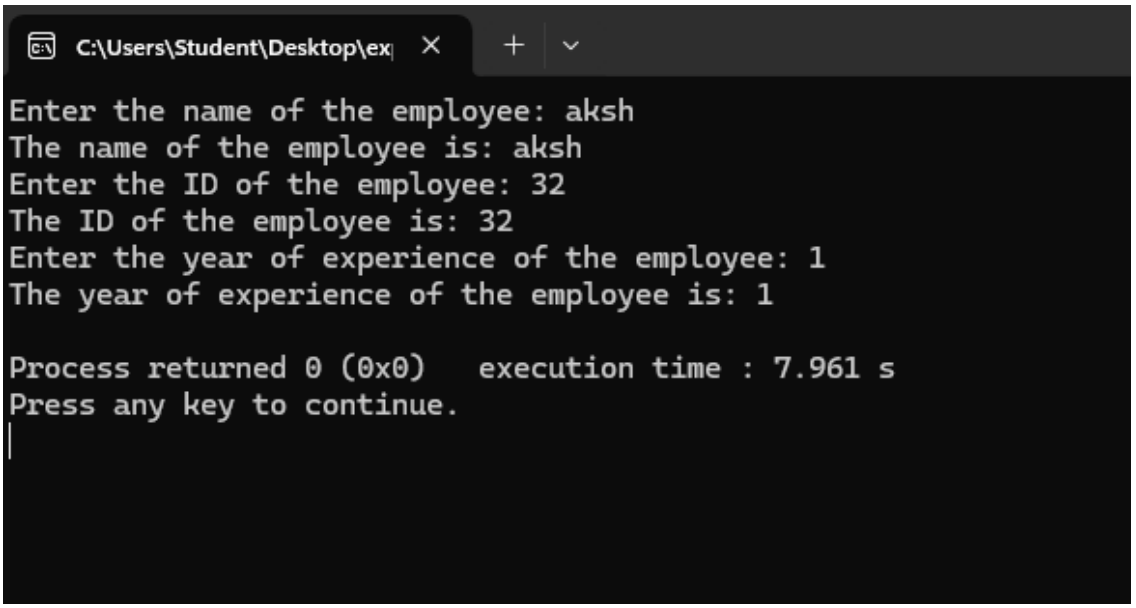
Output

```
Enter details for student 1
Enter roll no: 3
Enter name of student:Aadya
Enter marks for student in percentage out of 100: 89.5
Enter details for student 2
Enter roll no: 4
Enter name of student:Aakanksha
Enter marks for student in percentage out of 100: 95.7
Enter details for student 3
Enter roll no: 19
Enter name of student:Aditi.K
Enter marks for student in percentage out of 100: 90
Enter details for student 4
Enter roll no: 20
Enter name of student:Aditi.S
Enter marks for student in percentage out of 100: 94.3
Enter details for student 5
Enter roll no: 26
Enter name of student:Aditya
Enter marks for student in percentage out of 100: 94.3
Enter details for student 6
Enter roll no: 31
Enter name of student:Akanksha
Enter marks for student in percentage out of 100: 80.0
Enter details for student 7
Enter roll no: 32
Enter name of student:Aksh
Enter marks for student in percentage out of 100: 95.0
Enter details for student 8
Enter roll no: 38
Enter name of student:Amrit
Enter marks for student in percentage out of 100: 83.3
Enter details for student 9
Enter roll no: 44
Enter name of student:Aniket
Enter marks for student in percentage out of 100: 93.8
Enter details for student 10
Enter roll no: 45
Enter name of student:Anish
Enter marks for student in percentage out of 100: 95.2
Details of students in decreasing order of percentage:
Roll No: 4, Name: Aakanksha, Percentage: 95.70
Roll No: 45, Name: Anish, Percentage: 95.20
Roll No: 32, Name: Aksh, Percentage: 95.00
Roll No: 20, Name: Aditi.S, Percentage: 94.30
Roll No: 26, Name: Aditya, Percentage: 94.30
Roll No: 44, Name: Aniket, Percentage: 93.80
Roll No: 19, Name: Aditi.K, Percentage: 90.00
Roll No: 3, Name: Aadya, Percentage: 89.50
Roll No: 38, Name: Amrit, Percentage: 83.30
Roll No: 31, Name: Akanksha, Percentage: 80.00

Process returned 0 (0x0)   execution time : 109.978 s
Press any key to continue.
```

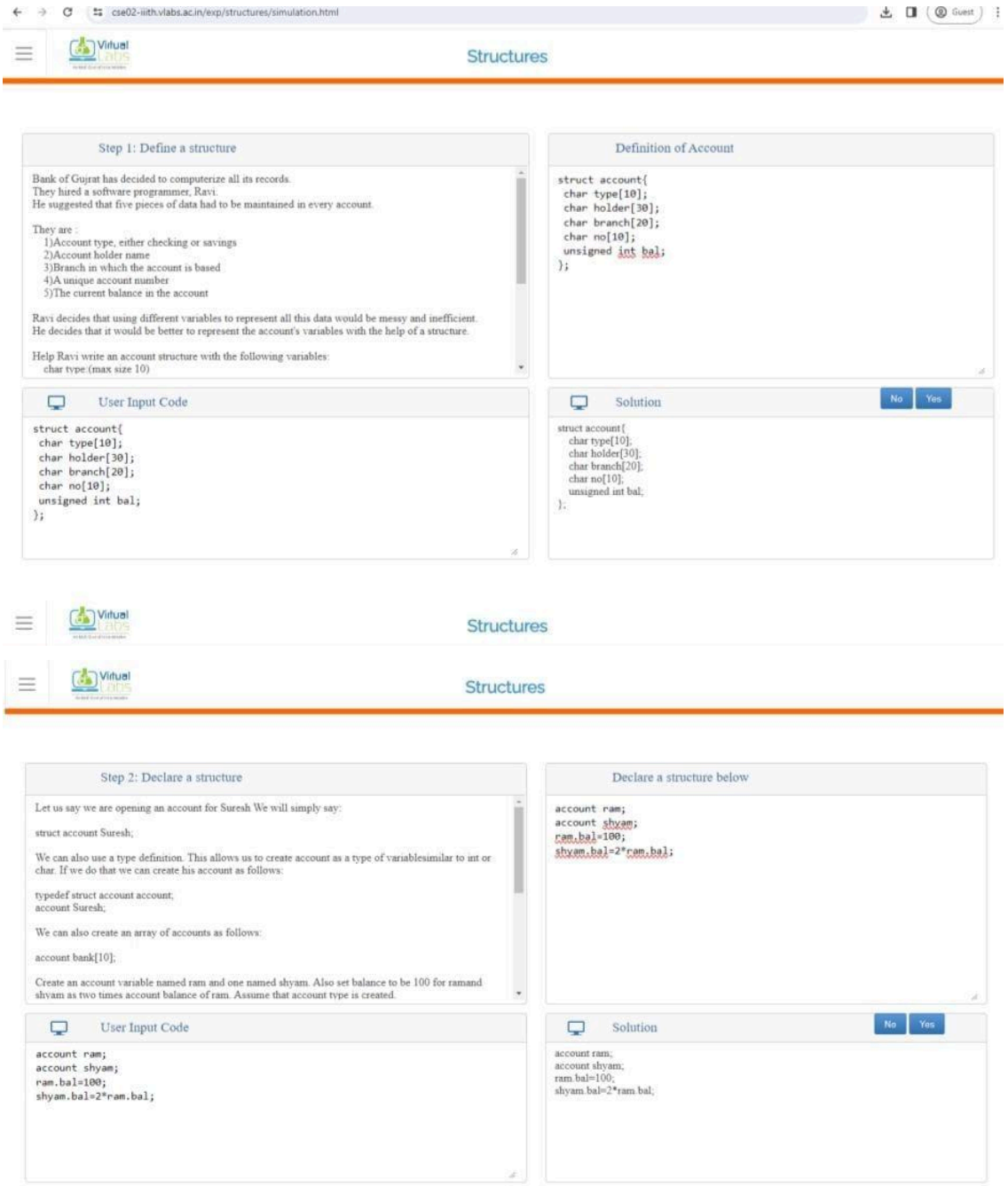
- WAP to display employee name, ID and year of experience using union.

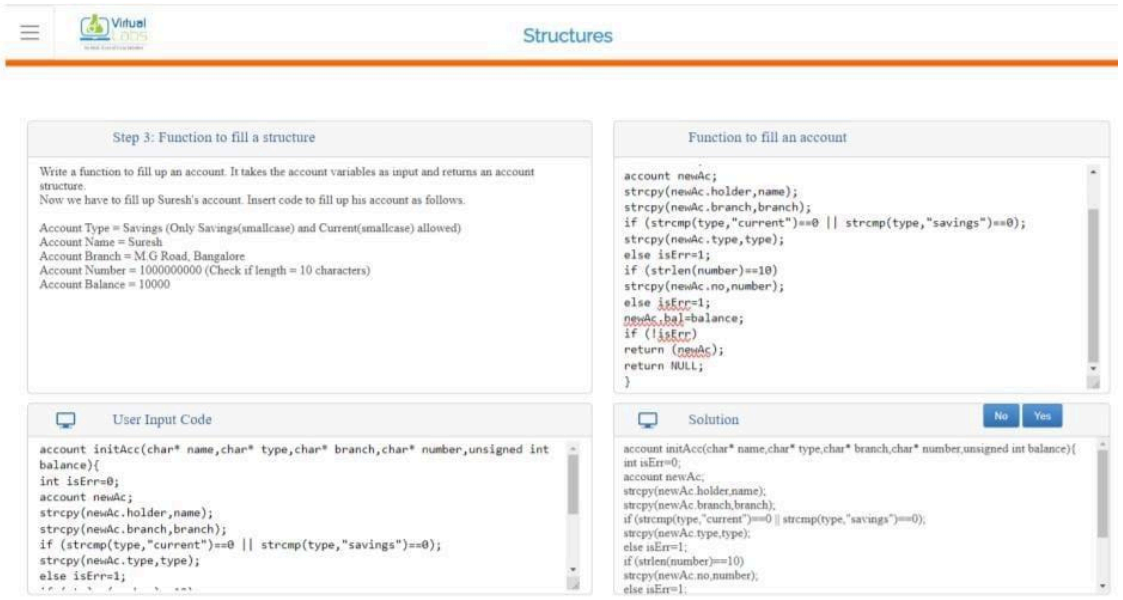
```
#include <stdio.h>
union employee
{
char name[20];
int id;
int year;
};
int main()
{
 union employee e;
 printf("Enter the name of the employee: ");
 scanf("%s",e.name);
 printf("The name of the employee is: %s\n",e.name);
 printf("Enter the ID of the employee: ");
 scanf("%d",&e.id);
 printf("The ID of the employee is: %d\n",e.id);
 printf("Enter the year of experience of the employee: ");
 scanf("%d",&e.year);
 printf("The year of experience of the employee is: %d\n",e.year);
 return 0;
}
```



- Virtual lab on Structure and Union

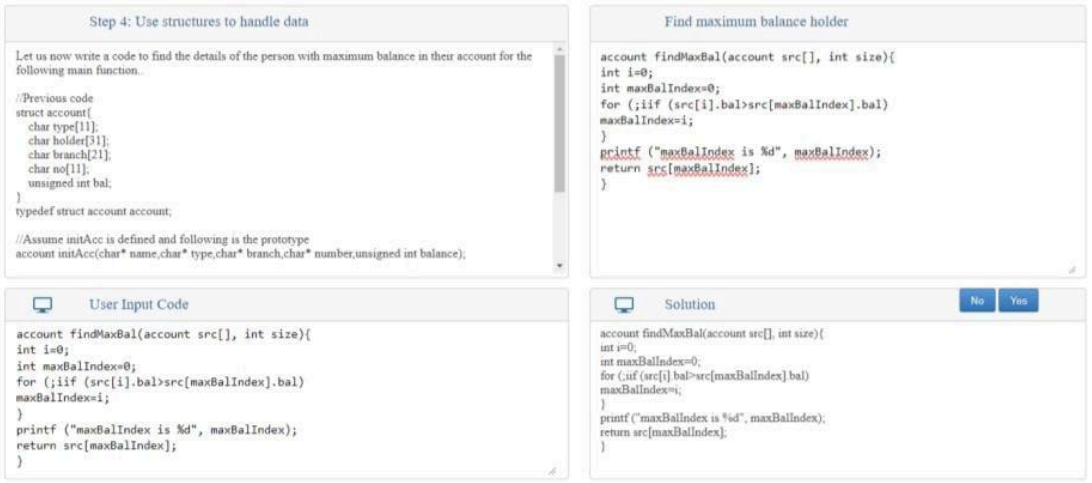
  https://cse02-iiith.vlabs.ac.in/exp/structures/simulation.html


cse02-iiith.vlabs.ac.in/exp/structures/simulation.html
Structures
Step 1: Define a structure
Bank of Gujrat has decided to computerize all its records.
They hired a software programmer, Ravi.
He suggested that five pieces of data had to be maintained in every account.
They are :
1)Account type, either checking or savings
2)Account holder name
3)Branch in which the account is based
4)A unique account number
5)The current balance in the account
Ravi decides that using different variables to represent all this data would be messy and inefficient.
He decides that it would be better to represent the account's variables with the help of a structure.
Help Ravi write an account structure with the following variables:
char type:(max size 10)
Definition of Account
struct account{
char type[10];
char holder[30];
char branch[20];
char no[10];
unsigned int bal;
};
User Input Code
struct account{
char type[10];
char holder[30];
char branch[20];
char no[10];
unsigned int bal;
};
Solution
No
Yes
struct account{
char type[10];
char holder[30];
char branch[20];
char no[10];
unsigned int bal;
};
Structures
Structures
Step 2: Declare a structure
Let us say we are opening an account for Suresh We will simply say:
struct account Suresh;
We can also use a type definition. This allows us to create account as a type of variablesimilar to int or char. If we do that we can create his account as follows:
typedef struct account account;
account Suresh;
We can also create an array of accounts as follows:
account bank[10];
Create an account variable named ram and one named shyam. Also set balance to be 100 for ramand shyam as two times account balance of ram. Assume that account type is created.
Declare a structure below
account ram;
account shyam;
ram.bal=100;
shyam.bal=2*ram.bal;
User Input Code
account ram;
account shyam;
ram.bal=100;
shyam.bal=2*ram.bal;
Solution
No
Yes
account ram;
account shyam;
ram.bal=100;
shyam.bal=2*ram.bal;


Structures
Step 3: Function to fill a structure
Write a function to fill up an account. It takes the account variables as input and returns an account structure.
Now we have to fill up Suresh's account. Insert code to fill up his account as follows.
Account Type = Savings (Only Savings(smallcase) and Current(smallcase) allowed)
Account Name = Suresh
Account Branch = M.G Road, Bangalore
Account Number = 1000000000 (Check if length = 10 characters)
Account Balance = 10000
Function to fill an account
account newAc;
strcpy(newAc.holder,name);
strcpy(newAc.branch,branch);
if (strcmp(type,"current")==0 || strcmp(type,"savings")==0);
strcpy(newAc.type,type);
else isErr=1;
if (strlen(number)==10)
strcpy(newAc.no,number);
else isErr=1;
newAc.bal=balance;
if (!isErr)
return (newAc);
return NULL;
}
User Input Code
account initAcc(char* name,char* type,char* branch,char* number,unsigned int balance){
int isErr=0;
account newAc;
strcpy(newAc.holder,name);
strcpy(newAc.branch,branch);
if (strcmp(type,"current")==0 || strcmp(type,"savings")==0);
strcpy(newAc.type,type);
else isErr=1;
Solution
No
Yes
account initAcc(char* name,char* type,char* branch,char* number,unsigned int balance){
int isErr=0;
account newAc;
strcpy(newAc.holder,name);
strcpy(newAc.branch,branch);
if (strcmp(type,"current")==0 || strcmp(type,"savings")==0);
strcpy(newAc.type,type);
else isErr=1;
if (strlen(number)==10)
strcpy(newAc.no,number);
else isErr=1;


Structures


Structures


Step 4: Use structures to handle data
Let us now write a code to find the details of the person with maximum balance in their account for the following main function.
//Previous code
struct account{
char type[11];
char holder[31];
char branch[21];
char no[11];
unsigned int bal;
}
typedef struct account account;
//Assume initAcc is defined and following is the prototype
account initAcc(char* name,char* type,char* branch,char* number,unsigned int balance);
Find maximum balance holder
account findMaxBal(account src[], int size){
int i=0;
int maxBalIndex=0;
for (;iif (src[i].bal>src[maxBalIndex].bal)
maxBalIndex=i;
}
printf ("maxBalIndex is %d", maxBalIndex);
return src[maxBalIndex];
}
User Input Code
account findMaxBal(account src[], int size){
int i=0;
int maxBalIndex=0;
for (;iif (src[i].bal>src[maxBalIndex].bal)
maxBalIndex=i;
}
printf ("maxBalIndex is %d", maxBalIndex);
return src[maxBalIndex];
}
Solution
No
Yes
account findMaxBal(account src[], int size){
int i=0;
int maxBalIndex=0;
for (;iif (src[i].bal>src[maxBalIndex].bal)
maxBalIndex=i;
}
printf ("maxBalIndex is %d", maxBalIndex);
return src[maxBalIndex];
}

Date: _____ **Signature of faculty in-charge**