TUTORIAL 7

1. Differentiate between structure and union?

	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

2. What is a member?

A structure contains an ordered group of data objects. Unlike the elements of an array, the data objects within a structure can have varied data types. Each data object in a structure is a member or field

Samuela Abigail
71762108039

3. How structure different from array?

ARRAY	STRUCTURE
Array refers to a collection consisting of elements of homogeneous data type.	Structure refers to a collection consisting of elements of heterogeneous data type.
Array uses subscripts or “[ ]” (square bracket) for element access	Structure uses “.” (Dot operator) for element access
Array is pointer as it points to the first element of the collection.	Structure is not a pointer
Instantiation of Array objects is not possible.	Instantiation of Structure objects is possible.
Array size is fixed and is basically the number of elements multiplied by the size of an element.	Structure size is not fixed as each element of Structure can be of different type and size.
Bit filed is not possible in an Array.	Bit filed is possible in an Structure.
Array declaration is done simply using [] and not any keyword.	Structure declaration is done with the help of “struct” keyword.
Arrays is a non-primitive datatype	Structure is a user-defined datatype.
Array traversal and searching is easy and fast.	Structure traversal and searching is complex and slow.
data_type array_name[size];	struct sruct_name{ data_type1 ele1; data_type2 ele2; };
Array elements are stored in continuous memory locations.	Structure elements may or may not be stored in a continuous memory location.
Array elements are accessed by their index number using subscripts.	Structure elements are accessed by their names using dot operator.

Samuela Abigail
71762108039

4. What are member, tag and variable name in structure and what purpose do they serve?

   A "structure declaration" names a type and specifies a sequence of variable values (called "members" or "fields" of the structure) that can have different types. An optional identifier, called a "tag," gives the name of the structure type and can be used in subsequent references to the structure type.

5. Difference between structure tag and structure instance?

A structure, an instance of a structure type, is a first-class value that contains a value for each field of the structure type. A structure instance is created with a type-specific constructor procedure, and its field values are accessed and changed with type-specific accessor and mutator procedures.

An optional identifier, called a "tag," gives the name of the structure type and can be used in subsequent references to the structure type. A variable of that structure type holds the entire sequence defined by that type.

6. Why can’t structures be compared?

   struct elements are usually aligned to some boundary, and when you initialize a struct (especially one on the stack), anything in the bytes skipped by alignment will be uninitialized. Moreover, the contents of n past the end of the constant initializer are not initialized.

7. How can two structures be compared?

Samuela Abigail
71762108039

To find out if they are the same object, compare pointers to the two structs for equality. If you want to find out in general if they have the same value you have to do a deep comparison. This involves comparing all the members. If the members are pointers to other structs you need to recurse into those structs too

8. Why do structures get padded?

In order to align the data in memory, one or more empty bytes (addresses) are inserted (or left empty) between memory addresses which are allocated for other structure members while memory allocation. This concept is called structure padding.

9. Explain the relationship of structure and pointer?

The pointers can be used to access the member of structures by declaring pointer to structure type

10. Explain the relationship of structure and function?

A structure information can be passed as a function arguments. The structure variable may be passed as a value or reference. The function will return the value by using the return statement.

11. Define ENUMERATION TYPES?

An enum type is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it.

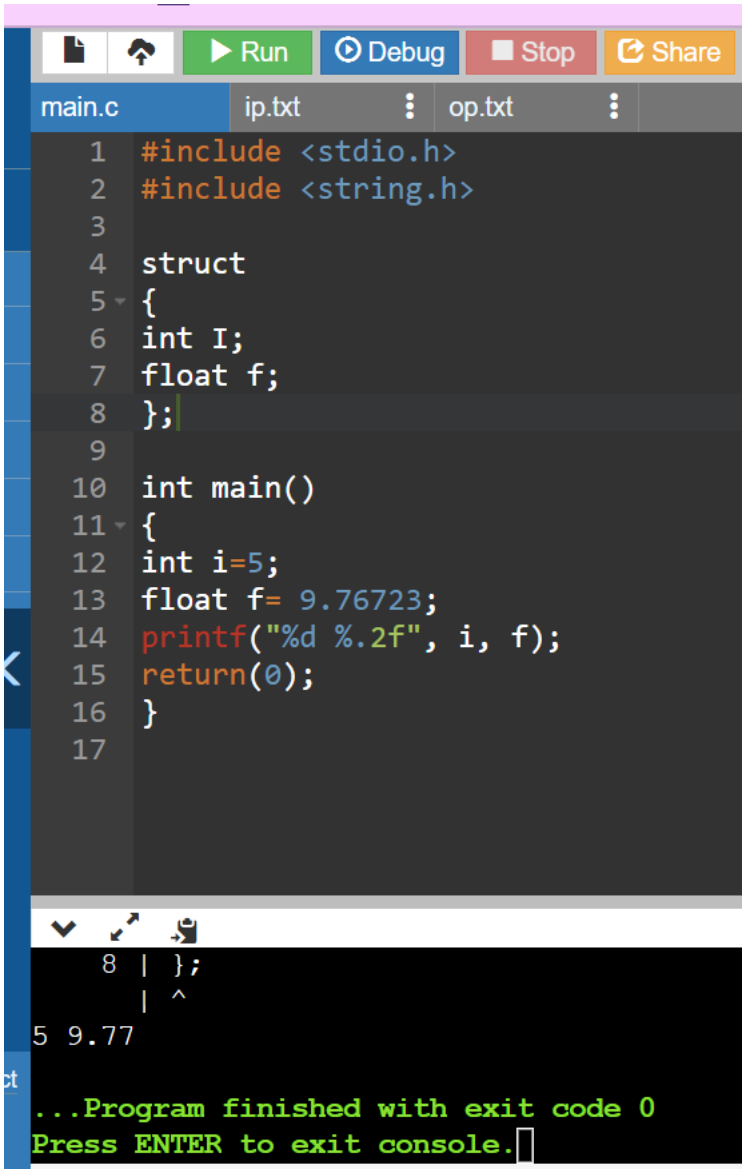
12. What will be the output?

```
struct
{
int I;
float f;
};
int main()
{
int i=5;
float f= 9.76723;
printf("%d %.2f ", I, f);
return(0)
}
```

It'll show error saying I is undeclared. Corrected output-

(it'll have sematic error since structure declaration doesn't have tag)

Samuela Abigail
71762108039



Samuela Abigail
71762108039

Practice!!

Declare a structure to store the following information of an employee:

- Employee code

- Employee name

- Salary

- Department number

- Date of joint (it itself a structure consisting of date, month and year)

Write a c program to store the data of ‘n’ employees where n is given by the user (use dynamic memory allocation). Include a menu that will allow user to select any of the following features:

a) Use a function to display employee information getting the maximum and minimum salary

b) Use a function to display employee records in ascending order according to their salary

c) Use a function to display the employee records in ascending order to their date of join

d) Use the function to display the department wise employee records

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>


struct employee
```

```
{
char code[30];
char name[50];
float salary;
int dep_num;

struct DOJ{
   int dd;
   int mm;
   int yy;
}doj;

}t;

void one(int n, struct employee *emp)
{
   int i,j;

   for(i=0;i<n;i++)
   {
      for(j=0;j<n-1;j++)
      {
         if((emp+j)->salary < (emp+j+1)->salary)//descending order
         {
```

```
            t=emp[j];
            emp[j]=emp[j+1];
            emp[j+1]=t;
          }
      }
   }

      printf("\nDetails of employee with maximum salary: ");
      printf("\nName: %s",emp->name);
      printf("\nCode: %s",emp->code);
      printf("\nSalary: %f",emp->salary);
      printf("\nDepartment number: %d",emp->dep_num);
      printf("\nDate of joining: %d/%d/%d",emp->doj.dd,emp->doj.mm,emp->doj.yy);

      printf("\n\nDetails of employee with minimum salary: ");
      printf("\nName: %s",(emp+n-1)->name);
      printf("\nCode: %s",(emp+n-1)->code);
      printf("\nSalary: %f",(emp+n-1)->salary);
      printf("\nDepartment number: %d",(emp+n-1)->dep_num);
      printf("\nDate of joining: %d/%d/%d",(emp+n-1)->doj.dd,(emp+n-1)->doj.mm,(emp+n-1)->doj.yy);
```

```
}

void two(int n, struct employee *emp)
{
   int i,j;

   for(i=0;i<n;i++)
   {
      for(j=0;j<n-1;j++)
      {
         if((emp+j)->salary < (emp+j+1)->salary)//descending order
         {
            t=emp[j];
            emp[j]=emp[j+1];
            emp[j+1]=t;
         }
      }
   }

   for(i=0;i<n;i++)
   {
      printf("\n\nEmployee %d: ",i+1);
      printf("\nName: %s",(emp+i)->name);
      printf("\nCode: %s",(emp+i)->code);
```

```
        printf("\nSalary: %f",(emp+i)->salary);
        printf("\nDepartment number: %d",(emp+i)->dep_num);
        printf("\nDate of joining: %d/%d/%d",(emp+i)->doj.dd,(emp+i)->doj.mm,(emp+i)->doj.yy);

    }



}

void three(int n, struct employee *emp)
{
    int i,j;

    for(i=0;i<n;i++)//newest to oldest employee
    {
        for(j=0;j<n-1;j++)
        {
            if((emp+j)->doj.yy < (emp+j+1)->doj.yy)//like 2014 is newest compared to 2004
            {
                t=emp[j];
                emp[j]=emp[j+1];
                emp[j+1]=t;
```

```
        }

        else if((emp+j)->doj.mm < (emp+j+1)->doj.mm && (emp+j)->doj.yy == (emp+j+1)->doj.yy)
        //like june (6) is newest compared to march (3)
        {
           t=emp[j];
           emp[j]=emp[j+1];
           emp[j+1]=t;
        }

        else if((emp+j)->doj.dd < (emp+j+1)->doj.dd &&//like 31 is newest compared to 14
        (emp+j)->doj.mm == (emp+j+1)->doj.mm &&
        (emp+j)->doj.yy == (emp+j+1)->doj.yy)
        {
            t=emp[j];
           emp[j]=emp[j+1];
           emp[j+1]=t;
        }
      }
   }

   for(i=0;i<n;i++)
```

```
    {
        printf("\n\nEmployee %d: ",i+1);
        printf("\nName: %s",(emp+i)->name);
        printf("\nCode: %s",(emp+i)->code);
        printf("\nSalary: %f",(emp+i)->salary);
        printf("\nDepartment number: %d",(emp+i)->dep_num);
        printf("\nDate of joining: %d/%d/%d",(emp+i)->doj.dd,(emp+i)->doj.mm,(emp+i)->doj.yy);

    }


}


void four(int n, struct employee *emp)
{
    int i,j;

    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1;j++)
        {
```

```
        if((emp+j)->dep_num > (emp+j+1)->dep_num)//ascending
order
        {
           t=emp[j];
           emp[j]=emp[j+1];
           emp[j+1]=t;
        }
      }
   }

   for(i=0;i<n;i++)
   {
      printf("\n\nEmployee %d: ",i+1);
      printf("\nName: %s",(emp+i)->name);
      printf("\nCode: %s",(emp+i)->code);
      printf("\nSalary: %f",(emp+i)->salary);
      printf("\nDepartment number: %d",(emp+i)->dep_num);
      printf("\nDate of joining: %d/%d/%d",(emp+i)->doj.dd,(emp+i)-
>doj.mm,(emp+i)->doj.yy);

   }


}
```

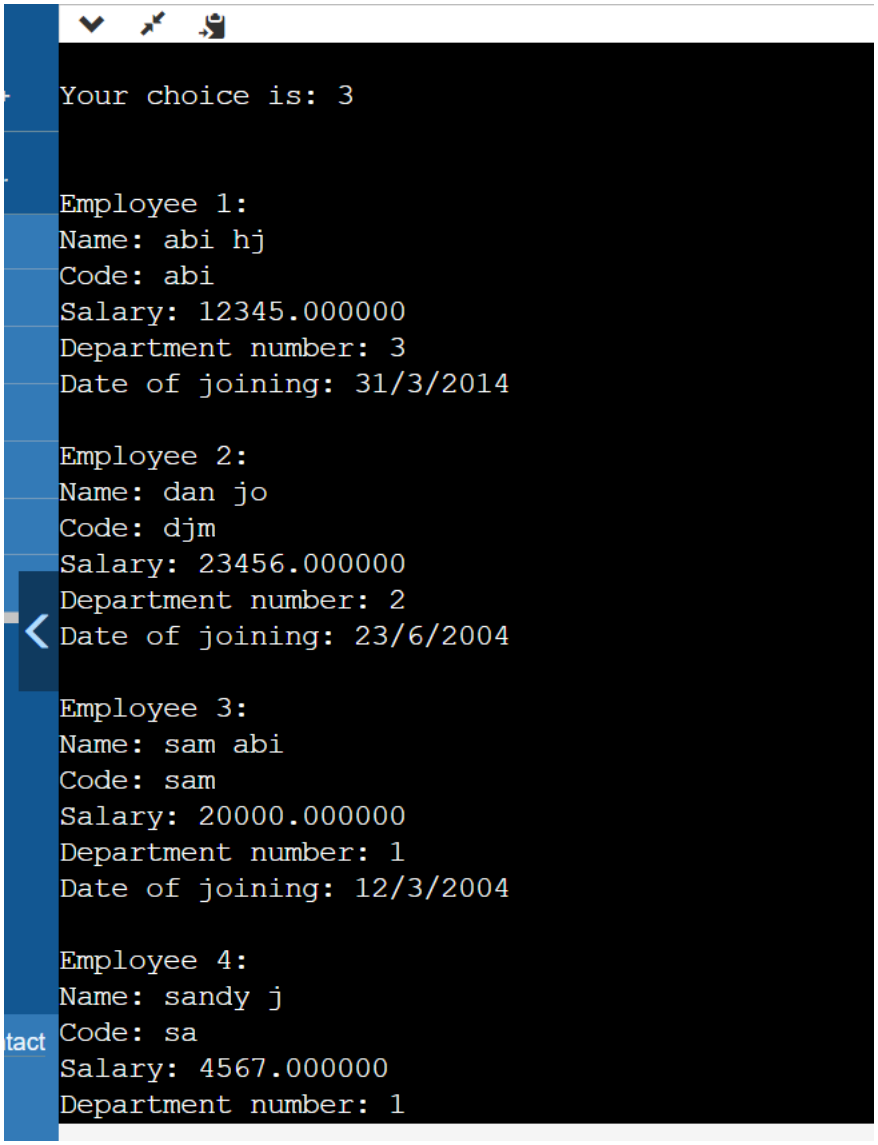
Samuela Abigail
71762108039

```
int main()
{
   struct employee *emp;
   int n, option, i;

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

    emp=(struct employee*)malloc(n*sizeof(struct
employee));//dynamic memory allocation

    for(i=0;i<n;i++)
    {
       printf("\nEnter details of employee %d: ",i+1);
       printf("\nName: ");
       getchar();
       scanf("%[^\n]s",(emp+i)->name);

       printf("\nCode: ");
       scanf("%s",(emp+i)->code);

       printf("\nSalary: ");
```

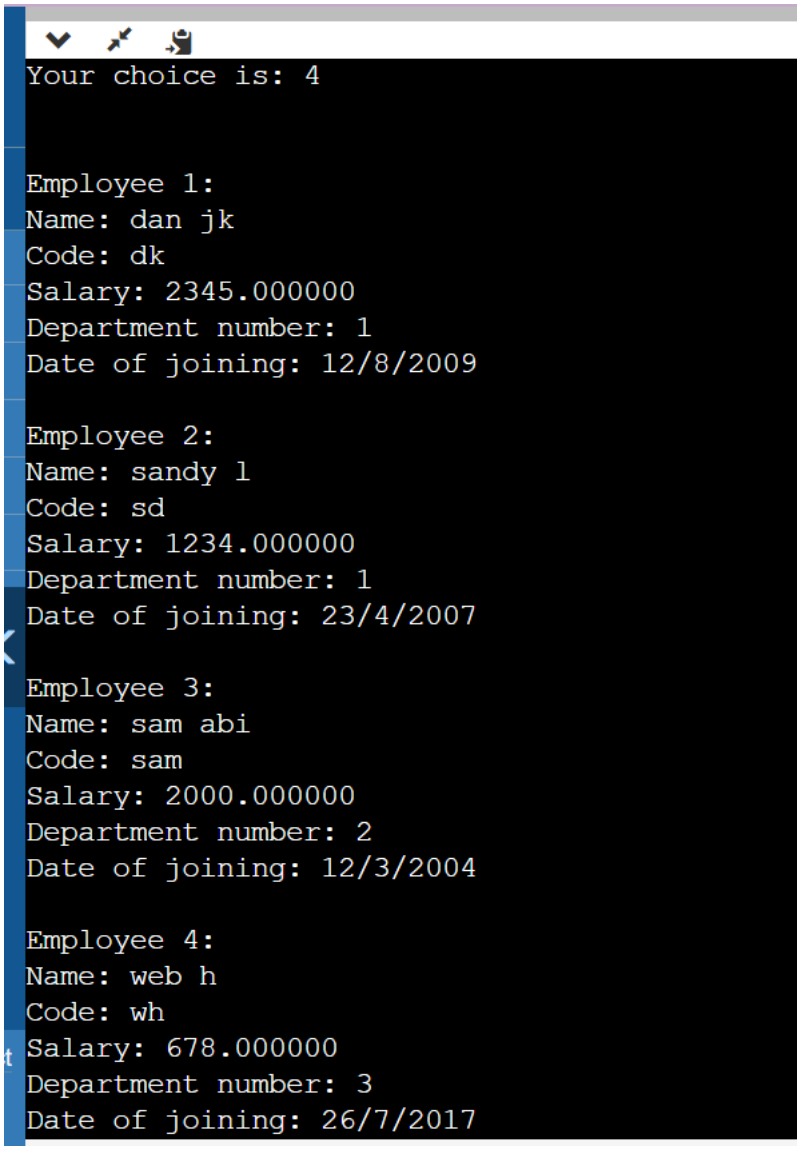
```
        scanf("%f",&(emp+i)->salary);

        printf("\nDepartment number: ");
        scanf("%d",&(emp+i)->dep_num);

        printf("\nDate of joining (dd/mm/yyyy): ");
        scanf("%d %d %d",&(emp+i)->doj.dd,&(emp+i)->doj.mm,&(emp+i)->doj.yy );

    }

    printf("\nSelect your choice: ");
    printf("\n1.display employee information getting the maximum and minimum salary");
    printf("\n2.display employee records in ascending order according to their salary");
    printf("\n3.display the employee records in ascending order to their date of join");
    printf("\n4.display the department wise employee records");
    printf("\n\nYour choice is: ");
    scanf("%d", &option);

    switch(option){
        case 1:
          one(n, emp);
```

```
            break;

        case 2:
            two(n, emp);
            break;

        case 3:
            three(n, emp);
            break;

        case 4:
            four(n, emp);
            break;

        default:
             printf("\nEnter correct option!");
             break;
    }

return 0;
}
```

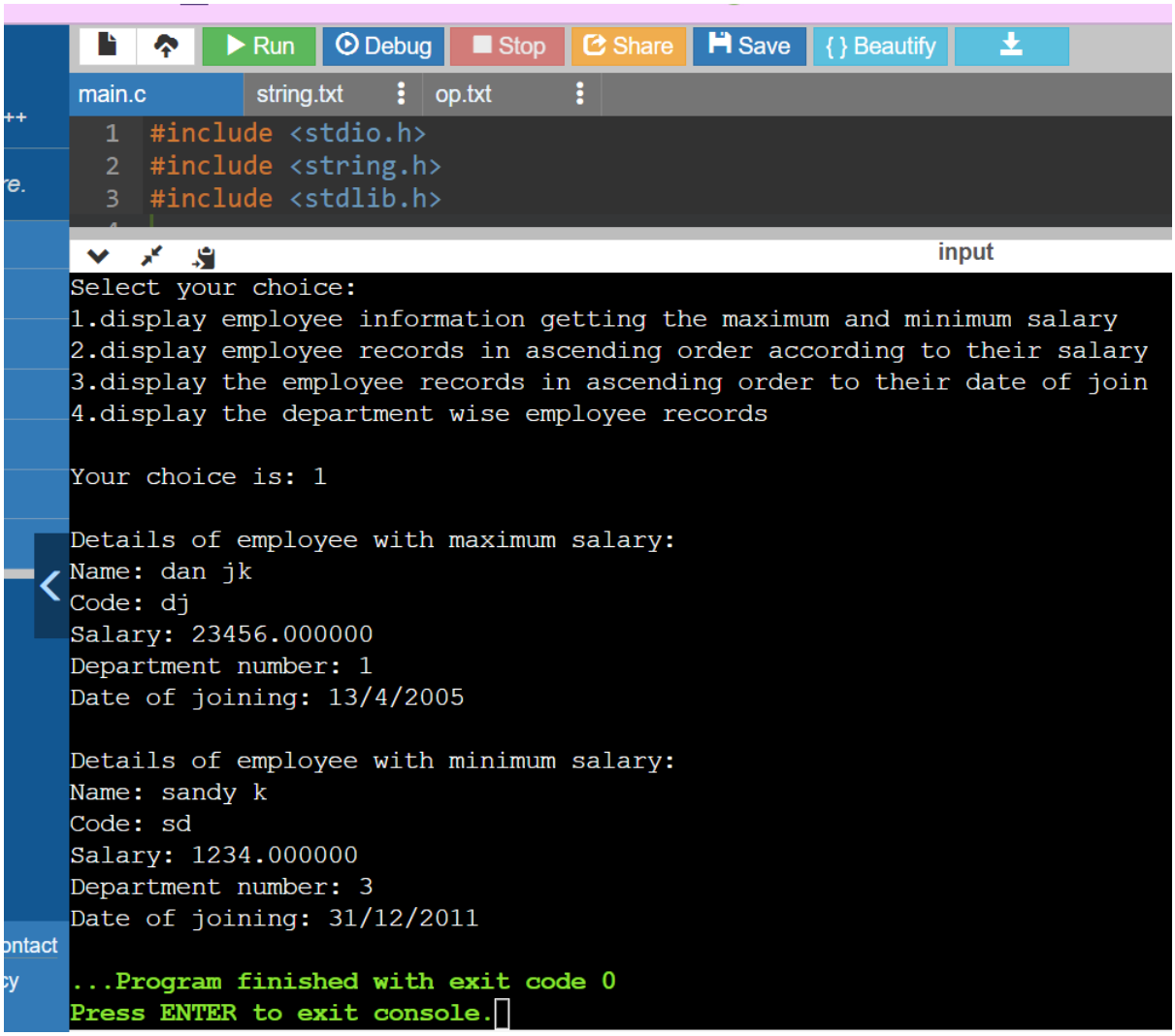
Samuela Abigail
71762108039



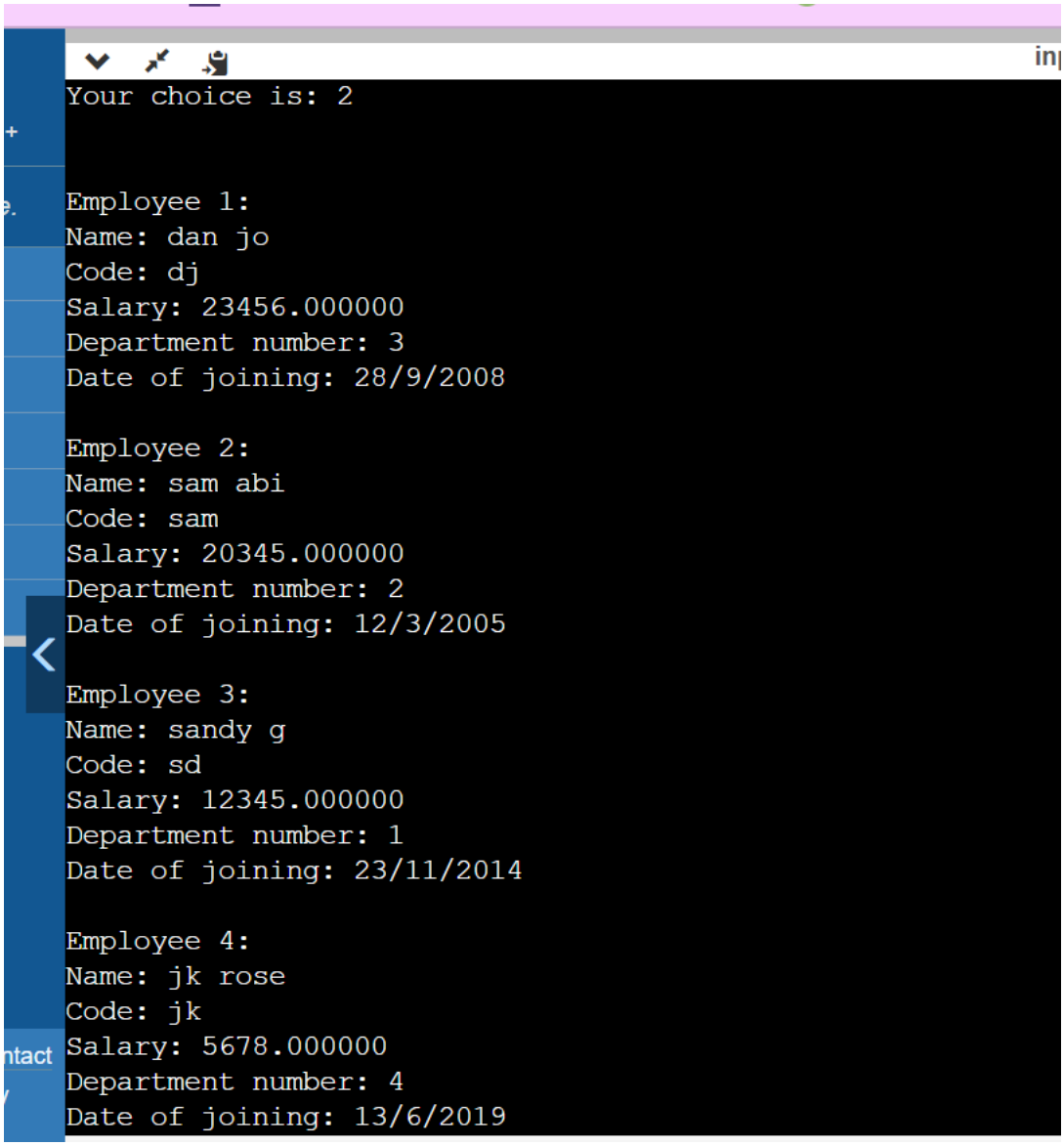
Samuela Abigail
71762108039



Samuela Abigail
71762108039



Samuela Abigail
71762108039



Samuela Abigail
71762108039