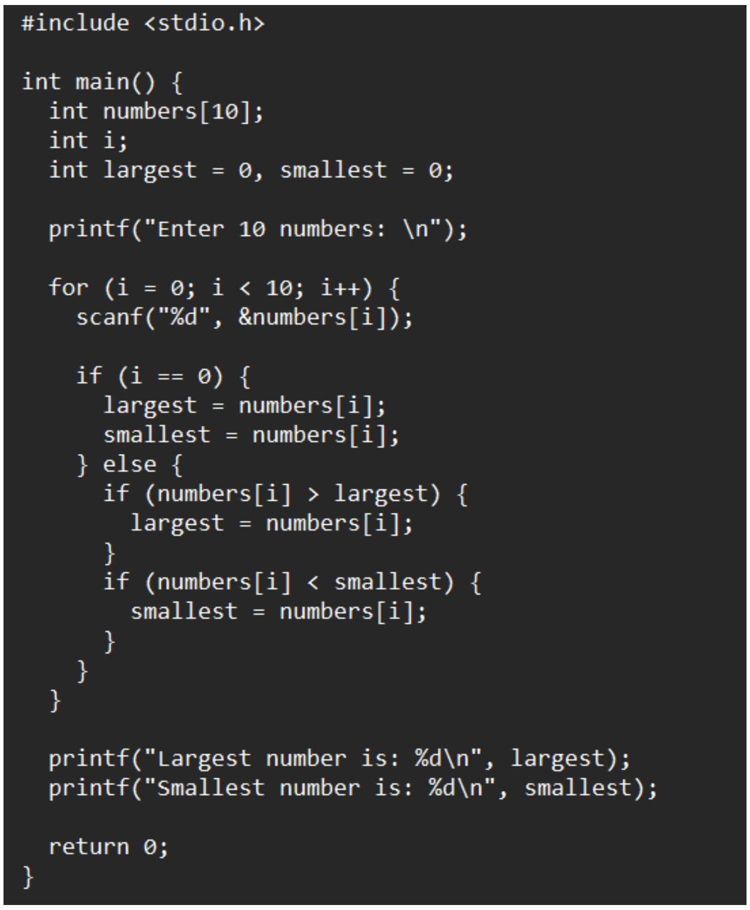
C programming assignment

1. Write the function of all unix commands.

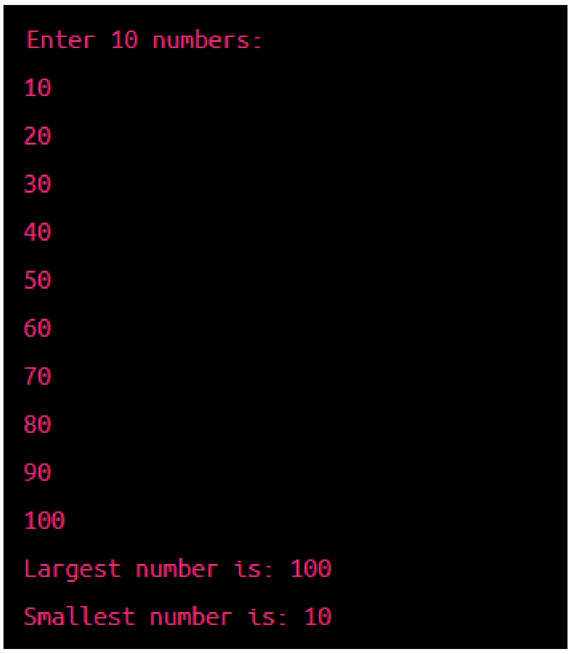
    1. `ls`: Lists the files and directories in the current directory.
    2. `cd`: Changes the current directory to the specified directory.
    3. `pwd`: Prints the current working directory.
    4. `mkdir`: Makes a new directory in the specified location.
    5. `rmdir`: Removes an empty directory.
    6. `cp`: Copies a file or directory to a specified location.
    7. `mv`: Renames a file or directory, or moves it to a different location.
    8. `rm`: Deletes a file or directory.
    9. `cat`: Displays the contents of a file.
    10. `less`: Shows the contents of a file one screen at a time.
    11. `head`: Shows the first few lines of a file.
    12. `tail`: Shows the last few lines of a file.
    13. `grep`: Searches for a specific pattern in a file.
    14. `find`: Searches for files or directories based on criteria such as name, size, or date modified.
    15. `sort`: Sorts the contents of a file.
    16. `wc`: Shows the number of lines, words, and characters in a file.
    17. `chmod`: Changes the permissions for a file or directory.
    18. `chown`: Changes the owner of a file or directory.

2. Write a program in C to find largest and smallest of the 10 numbers using array.



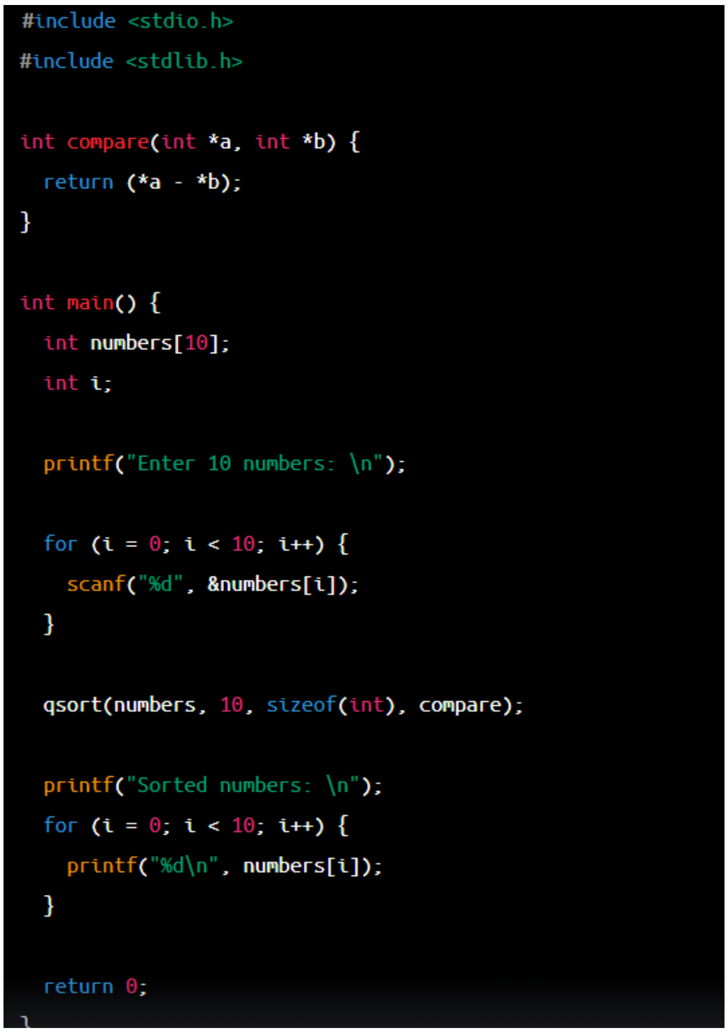
```c
#include <stdio.h>

int main() {
  int numbers[10];
  int i;
  int largest = 0, smallest = 0;

  printf("Enter 10 numbers: \n");

  for (i = 0; i < 10; i++) {
    scanf("%d", &numbers[i]);

    if (i == 0) {
      largest = numbers[i];
      smallest = numbers[i];
    } else {
      if (numbers[i] > largest) {
        largest = numbers[i];
      }
      if (numbers[i] < smallest) {
        smallest = numbers[i];
      }
    }
  }

  printf("Largest number is: %d\n", largest);
  printf("Smallest number is: %d\n", smallest);

  return 0;
}
```

OUTPUT

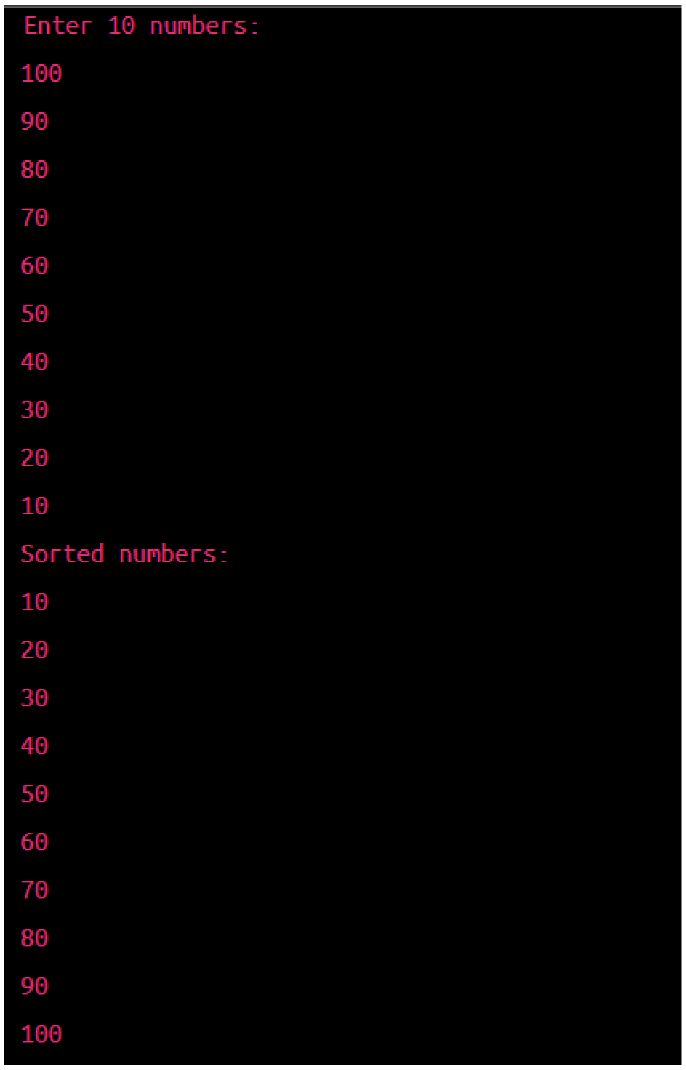


```
Enter 10 numbers:
10
20
30
40
50
60
70
80
90
100
Largest number is: 100
Smallest number is: 10
```

3. Write a program to shot the number using instruction short.

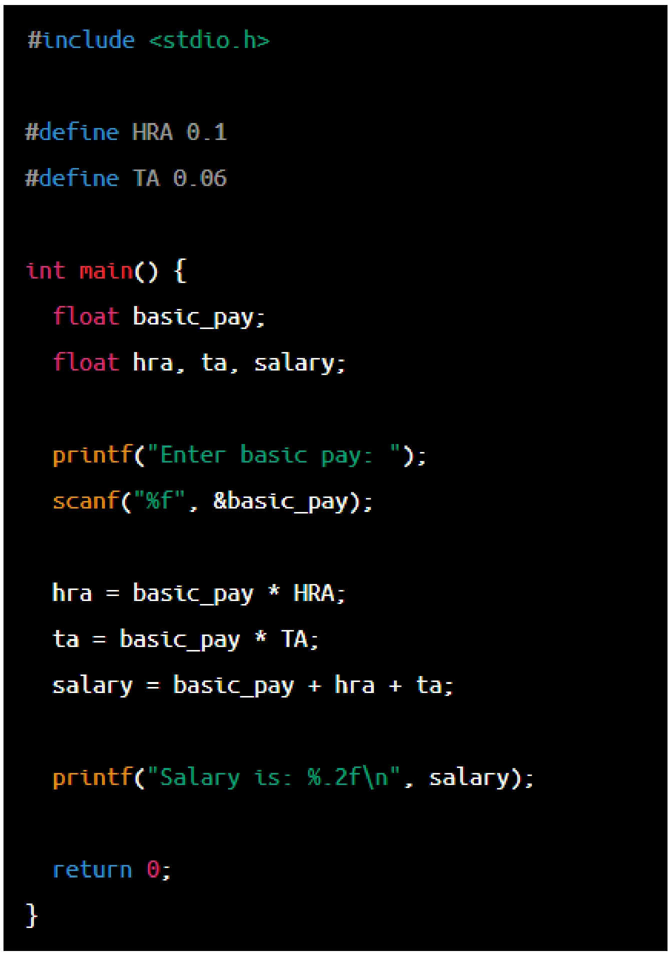

```
#include <stdio.h>
#include <stdlib.h>

int compare(int *a, int *b) {
  return (*a - *b);
}

int main() {
  int numbers[10];
  int i;

  printf("Enter 10 numbers: \n");

  for (i = 0; i < 10; i++) {
    scanf("%d", &numbers[i]);
  }

  qsort(numbers, 10, sizeof(int), compare);

  printf("Sorted numbers: \n");
  for (i = 0; i < 10; i++) {
    printf("%d\n", numbers[i]);
  }

  return 0;
```

OUTPUT



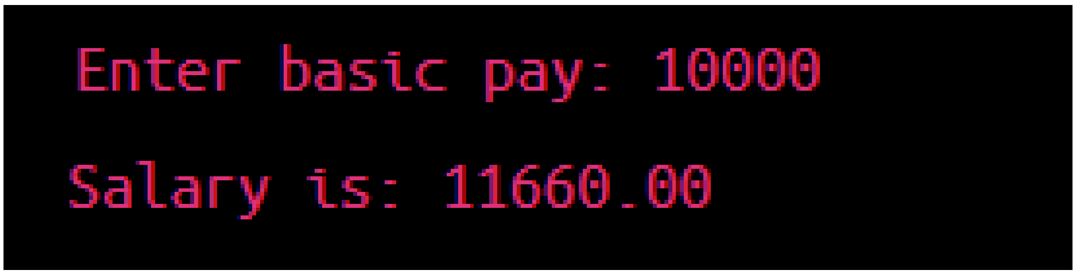
```
Enter 10 numbers:
100
90
80
70
60
50
40
30
20
10
Sorted numbers:
10
20
30
40
50
60
70
80
90
100
```

4. WAP to calculate the salary of the employee, where HRA= 10% of the basic pay, TA= 6%. Define HRA and TA as constant and use them to calculate the salary of the employees (Basic pay should be entered by user).



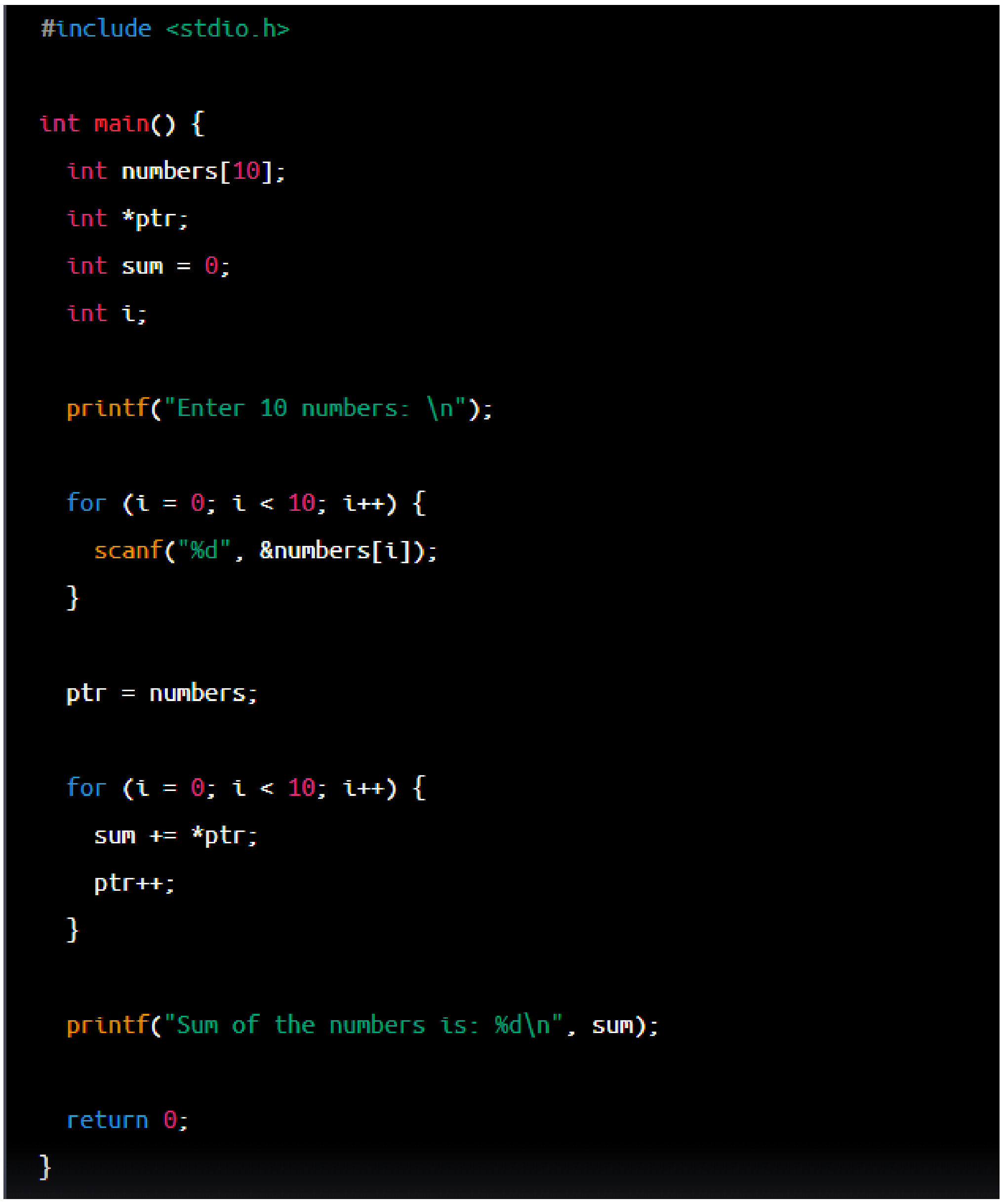
```c
#include <stdio.h>

#define HRA 0.1
#define TA 0.06

int main() {
  float basic_pay;
  float hra, ta, salary;

  printf("Enter basic pay: ");
  scanf("%f", &basic_pay);

  hra = basic_pay * HRA;
  ta = basic_pay * TA;
  salary = basic_pay + hra + ta;

  printf("Salary is: %.2f\n", salary);

  return 0;
}
```

OUTPUT



```
Enter basic pay: 10000
Salary is: 11660.00
```

5. WAP to find the sum of ten numbers using pointer. Write the program to explain bubble sort using array



```
#include <stdio.h>

int main() {
  int numbers[10];
  int *ptr;
  int sum = 0;
  int i;

  printf("Enter 10 numbers: \n");

  for (i = 0; i < 10; i++) {
    scanf("%d", &numbers[i]);
  }

  ptr = numbers;

  for (i = 0; i < 10; i++) {
    sum += *ptr;
    ptr++;
  }

  printf("Sum of the numbers is: %d\n", sum);

  return 0;
}
```

```
#include <stdio.h>
void bubble_sort(int arr[], int n) {
 int i, j;
 for (i = 0; i < n - 1; i++) {
  for (j = 0; j < n - i - 1; j++) {
   if (arr[j] > arr[j + 1]) {
    int temp = arr[j];
    arr[j] = arr[j + 1];
```

```
        arr[j + 1] = temp;
      }
    }
  }
}
int main() {
  int numbers[10];
  int i;
  printf("Enter 10 numbers: \n");
  for (i = 0; i < 10; i++) {
    scanf("%d", &numbers[i]);
  }
  bubble_sort(numbers, 10)
  printf("Sorted numbers: \n");
  for (i = 0; i < 10; i++) {
    printf("%d\n", numbers[i]);
  }
  return 0;
}
```

This program first reads ten numbers from the user and stores them in an array. Then, it calls the bubble_sort function, which sorts the array in ascending order using the bubble sort algorithm. Finally, the program outputs the sorted numbers.

6. Explain the call by address technique of passing parameters to a function?

The call by address technique of passing parameters to a function in C is a way of passing parameters to a function by passing the memory address of the parameters instead of their values. This allows the function to modify the values of the parameters, which will persist after the function returns.

When passing parameters by address, the function takes a pointer to the parameters as arguments, instead of the parameters themselves. The function then uses the dereference operator * to access the values of the parameters.

Here's an example of a program in C that demonstrates the call by address technique:

```
#include <stdio.h>

void swap(int *a, int *b) {

  int temp = *a;

  *a = *b;

  *b = temp;

}

int main() {

  int x = 10;

  int y = 20;

  printf("Before swapping: x = %d, y = %d\n", x, y);

  swap(&x, &y);

  printf("After swapping: x = %d, y = %d\n", x, y);

  return 0;

}
```

7. Is it possible to create an array of structure explain with the help of example

Yes, it is possible to create an array of structures in C. An array of structures is a collection of structures that have the same type. Each element in the array is a separate structure, and the array can be accessed using the index of the element.

Here's an example of a program in C that demonstrates how to create an array of structures:

```
#include <stdio.h>

struct person {

    char name[50];

    int age;

};

int main() {

    struct person people[3];

    for (int i = 0; i < 3; i++) {

        printf("Enter name: ");

        scanf("%s", people[i].name);

        printf("Enter age: ");

        scanf("%d", &people[i].age);

    }

    printf("\nDisplaying Information:\n");

    for (int i = 0; i < 3; i++) {

        printf("Name: %s\n", people[i].name);

        printf("Age: %d\n", people[i].age);

    }

    return 0;

}
```