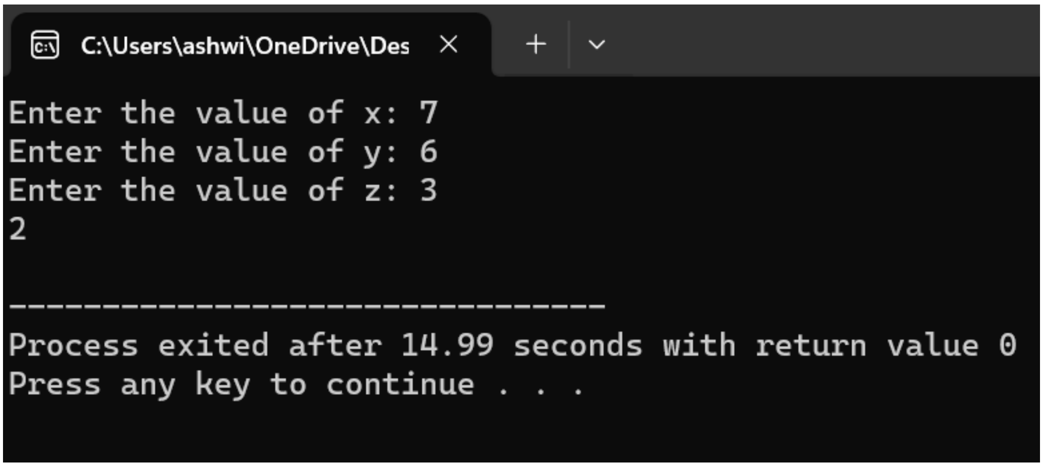
1. **Get three values x, y, z** and write a program to print 1 if x is the middle value, 2 if y is the middle value and 3 if z is the **middle value**. Assume that all three variables (x, y, z) are distinct and have different values

Code:

```
#include <stdio.h>

int main() {
  int x, y, z;

  printf("Enter the value of x: ");
  scanf("%d", &x);

  printf("Enter the value of y: ");
  scanf("%d", &y);

  printf("Enter the value of z: ");
  scanf("%d", &z);

  if ((x > y && x < z) || (x < y && x > z)) {
    printf("1\n");
  } else if ((y > x && y < z) || (y < x && y > z)) {
    printf("2\n");
  } else {
    printf("3\n");
  }

  return 0;
}
```



```
Enter the value of x: 7
Enter the value of y: 6
Enter the value of z: 3
2

--------------------------------
Process exited after 14.99 seconds with return value 0
Press any key to continue . . .
```

2. A password is said to be strong if it satisfies the following criteria:

It contains at least one lowercase English character.

It contains at least one uppercase English character.

It contains at least one special character.

The special characters are: !@#$%^&*()-+

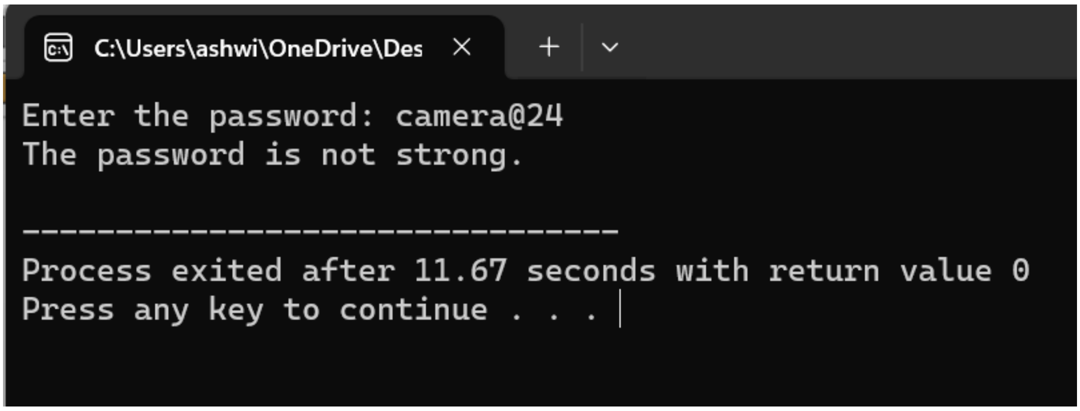
Its length is at least 8.

It contains at least one digit. **Given a string, find its strength.**

Code:

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

int isLowerCase(char c) {
   return (c >= 'a' && c <= 'z');
}

int isUpperCase(char c) {
   return (c >= 'A' && c <= 'Z');
}

int isDigit(char c) {
   return (c >= '0' && c <= '9');
}

int isSpecialCharacter(char c) {
   char specialChars[] = "!@#$%^&*()-+";
   for (int i = 0; i < strlen(specialChars); i++) {
      if (c == specialChars[i]) {
         return 1;
      }
   }
   return 0;
}

int isStrongPassword(char password[]) {
   int len = strlen(password);
   int hasLowerCase = 0; // false
   int hasUpperCase = 0; // false
   int hasDigit = 0;     // false
   int hasSpecialChar = 0; // false

   if (len < 8) {
        return 0; // false, Password length is less than 8
    }

    for (int i = 0; i < len; i++) {
        if (isLowerCase(password[i])) {
            hasLowerCase = 1; // true
        } else if (isUpperCase(password[i])) {
            hasUpperCase = 1; // true
        } else if (isDigit(password[i])) {
            hasDigit = 1; // true
        } else if (isSpecialCharacter(password[i])) {
            hasSpecialChar = 1; // true
        }
    }

    return (hasLowerCase && hasUpperCase && hasDigit && hasSpecialChar);
}

int main() {
    char password[50];

    printf("Enter the password: ");
    scanf("%s", password);

    if (isStrongPassword(password)) {
        printf("The password is strong.\n");
    } else {
        printf("The password is not strong.\n");
    }

    return 0;
}
```

Output:



```
Enter the password: camera@24
The password is not strong.

--------------------------------
Process exited after 11.67 seconds with return value 0
Press any key to continue . . .
```

3. A firm creates projects for which a certain number of hours are needed. The firm has a certain number of days. During 10% of the days, the workers are being trained and cannot work on the project. A normal working day is 8 hours long. The project is important for the firm and every worker must work on it with overtime of 2 hours per day. The hours must be rounded down to the nearest integer (for example, 6.98 hours are rounded to 6 hours). Write a program that calculates whether the firm can finish the project on time and how many hours more are needed or left.

Input:

Accept three integers as input(total number of hours needed,number of days,number of workers).

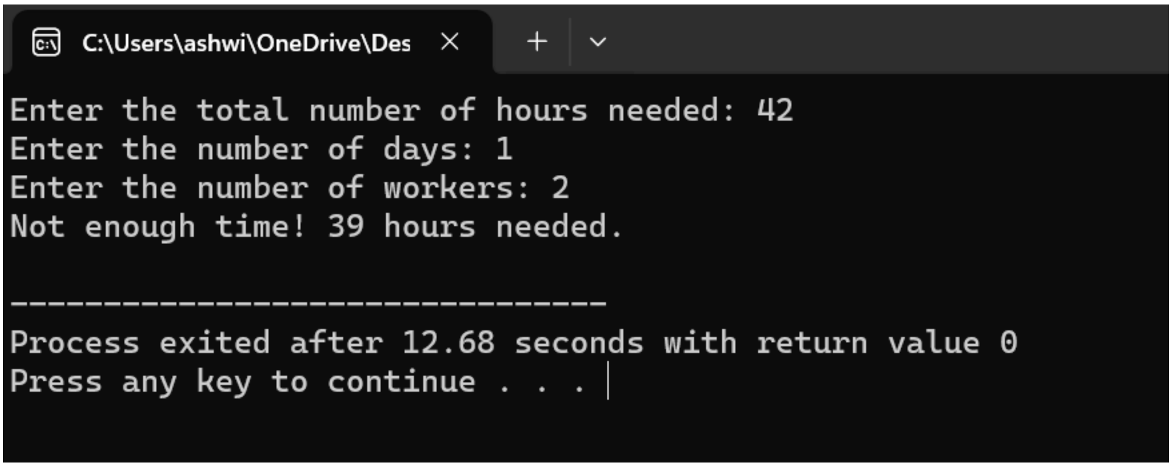
Output:

If the time is enough,print "Yes!{the hours left} hours left.".
If the time is NOT enough, print "Not enough time!{additional hours} hours needed.

Code:

```
#include <stdio.h>

int main() {
  int totalHoursNeeded, numberOfDays, numberOfWorkers;

  printf("Enter the total number of hours needed: ");
  scanf("%d", &totalHoursNeeded);

  printf("Enter the number of days: ");
  scanf("%d", &numberOfDays);

  printf("Enter the number of workers: ");
  scanf("%d", &numberOfWorkers);

  int totalWorkingHours = numberOfDays * 8 * 0.9;

  int totalProjectHours = totalHoursNeeded + numberOfDays * numberOfWorkers * 2;

  if (totalWorkingHours >= totalProjectHours) {
    int hoursLeft = totalWorkingHours - totalProjectHours;
    printf("Yes! %d hours left.\n", hoursLeft);
  } else {
    int additionalHours = totalProjectHours - totalWorkingHours;
    printf("Not enough time! %d hours needed.\n", additionalHours);
  }
```

```
    return 0;
}
```



```
Enter the total number of hours needed: 42
Enter the number of days: 1
Enter the number of workers: 2
Not enough time! 39 hours needed.

--------------------------------
Process exited after 12.68 seconds with return value 0
Press any key to continue . . .
```