Project 2, Program Design

1. (60 points) Slot machines are among the most popular attractions at land-based casinos and online casinos alike. They use random number generators to randomize all outcomes to ensure that every spin generates a result that is independent of the previous spin. Write a C program that simulates a slot machine that takes initial balance of the player and display the remaining balance after spinning. Assume that there are 14 symbols with internal representation of the slot machine.

Symbol	**Internal Storage**
Pineapple	0
Kiwi	1
Apple	2
Orange	3
Lime	4
Peach	5
Lemon	6
Pear	7
Banana	8
Cherry	9
Grape	10
Blueberry	11
Blackberry	12
Apricot	13

Your program should start with an initial balance (in cents) entered by the user. The program then allows the user to play a spin by randomly generating 3 numbers in the range (0-13) and print out the results for the spin. Each spin cost 5 cents. The user can keep spinning by pressing 1 or quit by pressing 0, or there is not sufficient balance. When there is a spin with 3 symbols being the same, the user win $1 (100 cents). When the program terminates, the remaining balance should be displayed.

Example run #1:

Enter initial balance (in cents): 7

Would you like to play (press 1 to play or 0 to quit)? 1

Output:

```
Blueberry Lime Kiwi
Remaining balance: 2 cents
Would you like to play (press 1 to play or 0 to quit)? 1
Output:
Insufficient fund
```

Example run #2:

```
Enter initial balance (in cents): 50
Would you like to play (press 1 to play or 0 to quit)? 1
Output:
Apricot Apple Grape
Remaining balance: 45 cents
Would you like to play (press 1 to play or 0 to quit)? 1
Output:
Apple Orange Lime
Remaining balance: 40 cents
Would you like to play (press 1 to play or 0 to quit)? 1
Output:
Lime Lime Lime
Congratulations! You have won $1
Remaining balance: 140 cents
Would you like to play (press 1 to play or 0 to quit)? 1
Output:
Banana Orange Banana
Remaining balance: 135 cents
Would you like to play (press 1 to play or 0 to quit)? 0
```

Output:

Thank you for playing!

Remaining balance: 135 cents

1) Name your program slot_machine.c.
2) Use rand() function to generate a random number. With the help of rand () function, a number in range of lower to upper can be generated as **num = (rand() % (upper – lower + 1)) + lower**
3) **rand() function generates the same sequence again and again every time the program runs. Use srand() function with time to set seed for rand() function so it generates different sequences of random numbers. Include the following statement at the beginning of the main function: srand(time(NULL));**
4) **To use the rand() and time function, you need to include <stdlib.h> and <time.h>.**

2. (40 points) Write a program to extract all integers from its input.

    1) Name your program extract_integer.c.
    2) The user input ends with the user pressing the enter key (a new line character).
    3) Library function isdigit() is not allowed for this program.
    4) Use getchar() to read in the input.

Example input/output:

Input: w8 4 me 2 finish 2!

Output: 8422

Before you submit:

1. Compile with –Wall. –Wall shows the warnings by the compiler. Be sure it compiles on ***student cluster*** with no errors and no warnings.

   gcc –Wall slot_machine.c

   gcc –Wall extract_integers.c

2. Be sure your Unix source file is read & write protected. Change Unix file permission on Unix:

   chmod 600 slot_machine.c

   chmod 600 extract_integers.c

3. Test your program extract_integers.c with the shell script on Unix:

   chmod +x try_digits

   ./try_digits

4. Download *slot_machine.c* and *extract_integers.c* from the student cluster and submit on Canvas.

Grading:

Total points: 100 (60 points problem 1 and 40 points problem 2)

1. A program that does not compile will result in a zero.
2. Runtime error and compilation warning 5%
3. Commenting and style 15%
4. Functionality requirement 80%

Programming Style Guidelines

The major purpose of programming style guidelines is to make programs easy to read and understand. Good programming style helps make it possible for a person knowledgeable in the application area to quickly read a program and understand how it works.

1. Your program should begin with a comment that briefly summarizes what it does. This comment should also include your **name**.
2. In most cases, a function should have a brief comment above its definition describing what it does. Other than that, comments should be written only *needed* in order for a reader to understand what is happening.
3. **Variable names** and function names should be sufficiently descriptive that a knowledgeable reader can easily understand what the variable means and what the function does. If this is not possible, comments should be added to make the meaning clear.

4. Use consistent **indentation** to emphasize block structure.
5. Full line comments inside function bodies should conform to the indentation of the code where they appear.
6. Macro definitions (#define) should be used for defining symbolic names for numeric constants. For example: **`#define PI 3.141592`**
7. Use names of moderate length for variables. Most names should be between 2 and 12 letters long.
8. Use either underscores or capitalization for compound names for variable: **`tot_vol`**, **`total_volumn`**, or **`totalVolumn`**.