Student Name: ROHIT
Student ID: 11607533
Roll No.: RK1654B40
Ques. No. Assigned: 11, 14
Email Address: rowhitswami1@gmail.com
GitHub Link: https://github.com/rowhitswami/OS-Assignment

1. Explain the problem in terms of operating system concept? (Max 200 word)

Problem 1: In this process one has to create a program to demonstrate the usage or 2 Zombie Processes and 2 Orphan Processes using System Calls. A process which has finished the execution but still has entry in the process table to report to its parent process is known as a **zombie process**. A child process always first becomes a zombie before being removed from the process table. The parent process reads the exit status of the child process which reaps off the child process entry from the process table.
A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an **orphan process**.

Problem 2: In the given problem of 10 students are purchasing the gift and returning to gift counter. The accountant gives the preference to that student who has maximum number of gifts. If we talk about this problem in Operating System concept, In priority scheduling algorithm, a priority is associated with each process and CPU is allocated to the process with the highest priority. If two processes have same priority then they will be execute in first come first serve order.

Code of Problem 1

```c
// Write a C program to create two zombie processes and two orphan processes using system calls.
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main()
{
    // Declaring and Forking Zombie Process
    pid_t zpid = fork();

    // Declaring and Forking Orphan Process
    pid_t opid = fork();

    // Parent process for zombie
    if (zpid > 0)
        sleep(5);

    // Child process for zombie
    else  {
        printf("(Zombie Process) Child exiting with process id %d because parent is sleeping.\n", getpid());
    }

    // Parent process for orphan
    if (opid > 0)
        printf("Parent Process with process id %d \n", getpid());

    // Child process for orphan process
    else if (opid == 0)
    {
        sleep(5);
        printf("(Orphan Process) Parent finishes execution and exits while the child process is still executing \n");
    }

    return 0;
}
```

Code of Problem 2 (I have implemented this problem in Python 3 language)

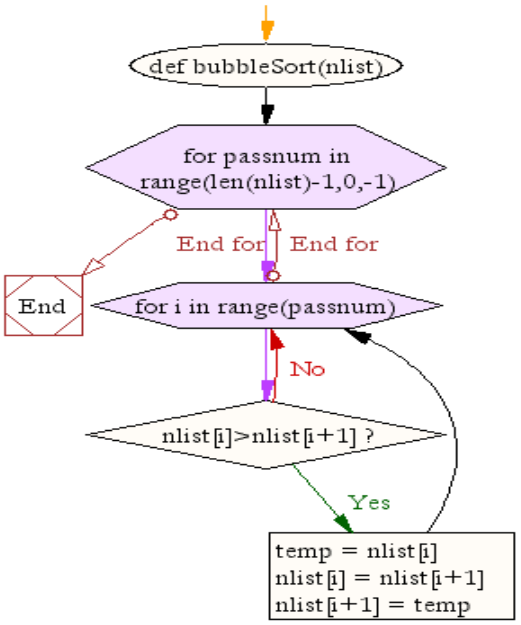
```
import random
from random import shuffle

# Sorting using Bubble Sort (Descending Order)
def bubbleSort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] < arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]

gift = []
# Populating list with random numbers (gifts)
gift = (random.sample(range(1, 50), 10))

# Calling Bubble Sort function
bubbleSort(gift)

# Creating students
students = []
s = "s"
students = [s+str(i) for i in range(1, 11)]

# Shuffling items in student (randomly reaching to the billing-counter)
shuffle(students)

# Zipping students with their gifts
mapping = list(zip(students, gift))

print("Order at which accountant gives preference to student who has maximum number of gifts")
print("Sr.\tStudent\t\tNo. of Gifts")
for i in range(10):
    print(i+1,"\t",mapping[i][0],"\t\t",mapping[i][1])
```

2. Write the algorithm for proposed solution of the assigned problem.
Problem 1 doesn't require any algorithm to implement.
I have used **Bubble Sort** algorithm in Problem 2 to sort the processes in descending order based on the arrival time of processes and updated there priority and sorted the order of process based on there current priority

3. Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity)

Step 1: Compare the first and the second element of the list and swap them if they are in wrong order. O(n)
Step 2: Compare the second and the third element of the list and swap them if they are in wrong order.
O(n)
Step 3: Proceed till the last element of the list in a similar fashion.
Step 4: Repeat all of the above steps until the list is sorted.

The complexity of bubble sort is **$O(n^2)$** in both worst and average cases, because the entire array needs to be iterated for every element.



4. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.
Problem 1: In this problem, we have to create 2 zombie processes and 2 orphan processes. Fork system call use for creates a new process, which is called child process, which runs concurrently with process (which process called system call fork) and this process is called parent process. After a new child process created, both processes will execute the next instruction following the fork() system call.

```
// Declaring and Forking Zombie Process
pid_t zpid = fork();

// Declaring and Forking Orphan Process
pid_t opid = fork();
```

Problem 2: In this problems, we are given constraint of 10 students. Whereas I applied a constraint of gifts a student can purchase, i.e. 1 <= gift <50
This implies that one has to buy at least 1 gift or no student can buy more than 49 gifts.

```
gift = []
# Populating list with random numbers (gifts)
gift = (random.sample(range(1, 50), 10))
```

5. If you have implemented any additional algorithm to support the solution, explain the need and usage of the same.
(Bubble Sort)
Bubble sort is (provably) the fastest sort available under a very specific circumstance. It originally became well known primarily because it was one of the first algorithms (of any kind) that was rigorously analyzed, and the proof was found that it was optimal under its limited circumstance.

```
1. begin BubbleSort(list)
2.      for all elements of list
3.         if list[i] > list[i+1]
4.             swap(list[i], list[i+1])
5.         end if
6.      end for
7.      return list
8. end BubbleSort
```

6. Explain the boundary conditions of the implemented code.
There are no boundary conditions for given problems.

7. Explain all the test cases applied on the solution of assigned problem.
Problem 1: This problem doesn't require any additional test case. We'll get a different output every-time when we execute the code of this problem. The screenshots of output are attached below.

Screenshots:
1st Execution

```
(Zombie Process) Child exiting with process id 8385 because parent is sleeping.
Parent Process with process id 8385
Parent Process with process id 8384
(Zombie Process) Child exiting with process id 8387 because parent is sleeping.
(Orphan Process) Parent finishes execution and exits while the child process is still executing
```

2nd Execution

```
(Zombie Process) Child exiting with process id 9560 because parent is sleeping.
Parent Process with process id 9560
Parent Process with process id 9559
(Zombie Process) Child exiting with process id 9562 because parent is sleeping.
(Orphan Process) Parent finishes execution and exits while the child process is still executing
```

3rd Execution

```
(Zombie Process) Child exiting with process id 10316 because parent is sleeping.
Parent Process with process id 10316
Parent Process with process id 10315
(Zombie Process) Child exiting with process id 10318 because parent is sleeping.
(Orphan Process) Parent finishes execution and exits while the child process is still executing
```

4th Execution

```
(Zombie Process) Child exiting with process id 11273 because parent is sleeping.
Parent Process with process id 11273
(Zombie Process) Child exiting with process id 11275 because parent is sleeping.
(Orphan Process) Parent finishes execution and exits while the child process is still executing
Parent Process with process id 11272
```

Problem 2: This problem doesn't require any additional test case. We'll get a different output every-time when we execute the code of this problem. The screenshots of output are attached below.

Screenshots:

1st Execution

```
Order at which accountant gives preference to student who has maximum number of gifts
Sr.     Student         No. of Gifts
1        s6              38
2        s8              37
3        s5              33
4        s4              32
5        s10             28
6        s7              26
7        s9              21
8        s2              18
9        s1              17
10       s3              6
```

2nd Execution

```
Order at which accountant gives preference to student who has maximum number of gifts
Sr.     Student         No. of Gifts
1        s2              44
2        s5              38
3        s7              33
4        s6              32
5        s8              28
6        s1              25
7        s9              23
8        s3              22
9        s10             17
10       s4              12
```

3rd Execution

```
Order at which accountant gives preference to student who has maximum number of gifts
Sr.     Student         No. of Gifts
1        s1              48
2        s7              42
3        s5              38
4        s3              26
5        s10             23
6        s2              18
7        s4              16
8        s9              15
9        s6              10
10       s8              3
```

4th Execution

```
Order at which accountant gives preference to student who has maximum number of gifts
Sr.     Student         No. of Gifts
1        s6              46
2        s10             35
3        s2              32
4        s5              30
5        s8              19
6        s4              17
7        s7              15
8        s9              14
9        s3              13
10       s1              3
```

8. Have you made minimum 5 revisions of solution on GitHub?
GitHub Link: https://github.com/rowhitswami/OS-Assignment