ASSIGNMENT - FUNCTIONS AND POINTERS

1. (**computePay**) Write a C function that determines the gross pay for an employee in a company. The company pays straight-time for the first 160 hours worked by each employee for four weeks and pays time-and-a-half for all hours worked in excess of 160 hours. The function takes in the number of hours each employee worked for the four weeks, and the hourly rate of each employee. Write the function in two versions. The function `computePay1()` returns the gross pay to the calling function, while the function `computePay2()` returns the gross pay to the calling function through the pointer parameter, `grossPay`. The function prototypes for the function are given as follows:

```
double computePay1(int noOfHours, int payRate);
void computePay2(int noOfHours, int payRate, double *grossPay);
```

A sample program template is given below to test the functions:

```
#include <stdio.h>
double computePay1(int noOfHours, int payRate);
void computePay2(int noOfHours, int payRate, double *grossPay);
int main()
{
   int noOfHours, payRate;
   double grossPay;

   printf("Enter number of hours: \n");
   scanf("%d", &noOfHours);
   printf("Enter hourly pay rate: \n");
   scanf("%d", &payRate);
   printf("computePay1(): %.2f\n", computePay1(noOfHours, payRate));
   computePay2(noOfHours, payRate, &grossPay);
   printf("computePay2(): %.2f\n", grossPay);
   return 0;
}
double computePay1(int noOfHours, int payRate)
{
   /* Write your code here */
}
void computePay2(int noOfHours, int payRate, double *grossPay)
{
   /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter number of hours:
176
Enter hourly pay rate:
6
computePay1(): 1104.00
computePay2(): 1104.00
```

(2) Test Case 2:
```
Enter number of hours:
34
Enter hourly pay rate:
6
computePay1(): 204.00
computePay2(): 204.00
```

(3) Test Case 3:
```
Enter number of hours:
156
```

```
Enter hourly pay rate:
8
computePay1(): 1248.00
computePay2(): 1248.00
```

2. (**countEvenDigits**) Write a C function to count the number of even digits, i.e. digits with values 0,2,4,6,8 in a non-negative number. For example, if `number` is 1234567, then 3 will be returned. Write the function in two versions. The function `countEvenDigits1()` returns the result, while the function `countEvenDigits2()` returns the result via the pointer parameter, `count`. The function prototypes are given below:

```
int countEvenDigits1(int number);
void countEvenDigits2(int number, int *count);
```

A sample program template is given below to test the functions:

```
#include <stdio.h>
int countEvenDigits1(int number);
void countEvenDigits2(int number, int *count);
int main()
{
   int number, result;

   printf("Enter a number: \n");
   scanf("%d", &number);
   printf("countEvenDigits1(): %d\n", countEvenDigits1(number));
   countEvenDigits2(number, &result);
   printf("countEvenDigits2(): %d\n", result);
   return 0;
}
int countEvenDigits1(int number)
{
   /* Write your code here */
}
void countEvenDigits2(int number, int *count)
{
   /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter a number:
1234567
countEvenDigits1(): 3
countEvenDigits2(): 3
```

(2) Test Case 2:
```
Enter a number:
2468
countEvenDigits1(): 4
countEvenDigits2(): 4
```

(3) Test Case 3:
```
Enter a number:
1357
countEvenDigits1(): 0
countEvenDigits2(): 0
```

3. (**allEvenDigits**) Write a function that returns either 1 or 0 according to whether or not all the digits of the positive integer argument number are even. For example, if the input argument is 2468, then the function should return 1; and if the input argument is 1234, then 0 should be returned. Write the function in two versions. The function `allEvenDigits1()` returns the result to the caller,

while `allEvenDigits2()` passes the result through the pointer parameter, `result`. The function prototypes are given below:

```
int allEvenDigits1(int num);
void allEvenDigits2(int num, int *result);
```

A sample program template is given below to test the functions:

```
#include  <stdio.h>
int allEvenDigits1(int num);
void allEvenDigits2(int num, int *result);
int main()
{
   int number, p=999, result=999;

   printf("Enter a number: \n");
   scanf("%d", &number);
   p = allEvenDigits1(number);
   if (p == 1)
      printf("allEvenDigits1(): yes\n");
   else if (p == 0)
      printf("allEvenDigits1(): no\n");
   else
      printf("allEvenDigits1(): error\n");
   allEvenDigits2(number, &result);
   if (result == 1)
      printf("allEvenDigits2(): yes\n");
   else if (result == 0)
      printf("allEvenDigits2(): no\n");
   else
      printf("allEvenDigits2(): error\n");
   return 0;
}
int allEvenDigits1(int num)
{
   /* Write your program code here */
}
void allEvenDigits2(int num, int *result)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter a number:
2468
allEvenDigits1(): yes
allEvenDigits2(): yes
```

(2) Test Case 2:
```
Enter a number:
1357
allEvenDigits1(): no
allEvenDigits2(): no
```

(3) Test Case 3:
```
Enter a number:
24
allEvenDigits1(): yes
allEvenDigits2(): yes
```

(4) Test Case 4:

```
Enter a number:
246
allEvenDigits1(): yes
allEvenDigits2(): yes
```

4. (**extEvenDigits**) Write a function that extracts the even digits from a positive number, and combines the even digits sequentially into a new number. The new number is returned to the calling function. If the input number does not contain any even digits, then the function returns -1. For example, if the input number is 1234567, then 246 will be returned; and if the input number is 13, then –1 will returned. Write the function in two versions. The function `extEvenDigits1()` returns the result to the caller, while the function `extEvenDigits2()` returns the result through the pointer parameter, `result`. The function prototypes are given as follows:

```c
int extEvenDigits1(int num);
void extEvenDigits2(int num, int *result);
```

A sample program template is given below to test the functions:

```c
#include <stdio.h>
#define INIT_VALUE 999
int extEvenDigits1(int num);
void extEvenDigits2(int num, int *result);
int main()
{
   int number, result = INIT_VALUE;

   printf("Enter a number: \n");
   scanf("%d", &number);
   printf("extEvenDigits1(): %d\n", extEvenDigits1(number));
   extEvenDigits2(number, &result);
   printf("extEvenDigits2(): %d\n", result);
   return 0;
}
int extEvenDigits1(int num)
{
   /* Write your program code here */
}
void extEvenDigits2(int num, int *result)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter a number:
1234
extEvenDigits1(): 24
extEvenDigits2(): 24
```

(2) Test Case 2:
```
Enter a number:
1357
extEvenDigits1(): -1
extEvenDigits2(): -1
```

(3) Test Case 3:
```
Enter a number:
2468
extEvenDigits1(): 2468
extEvenDigits2(): 2468
```

(4) Test Case 4:
```
Enter a number:
6
extEvenDigits1(): 6
extEvenDigits2(): 6
```