Homework 5

Problem 1

Program 1

```
/*----------------------------------------
Module name: program_1.c
Description: Comparison between prefix shortcut autodecrement
operator & postfix shortcut autodecrement operator
Author: Bo Yue
Rev. 0 05 Aug 2019
----------------------------------------*/

#include <stdio.h>
```

```c
//declaration of function decPre() & decPost()
int decPre(int myNum);
int decPost(int myNum);

int main()
{
    int pre   = 0;//store the result of prefix
    int post  = 0;//store the result of postfic
    int myNum = 5;//the variable

    pre  = decPre(myNum);
    post = decPost(myNum);

    printf("The result of (--5) is: %d\n", pre);
    printf("The result of (5--) is: %d", post);

    return 0;
}

//definition of decPre()
int decPre(int myNum)
{
    return (--myNum);
}

//definition of decPost()
int decPost(int myNum)
{
    return (myNum--);
}
```

Test Case

```
The result of (--5) is: 4
The result of (5--) is: 5
```

- The result of the two functions are different. For decPre(), the result is 4, while for decPost(), the result is 5.
- The **reason** for the difference lies in that the prefix shortcut autodecrement operator before a specific number **returns the decremented number**, while the postfix shortcut autodecrement operator before a specific number **returns the number** and then decrement the number.

Characteristics and Side Effects

- The program is simple and clear, and passes by value.

Problem 2

Program 2

```
/*----------------------------------------
Module name: program_2.c
Description: The program expands each TAB character in an input set
into corresponding contiguous sets of space characters.
Assume that a TAB is 5 characters wide.
Author: Bo Yue
Rev. 0 06 Aug 2019
----------------------------------------*/

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

int main()
{
    char rec[100] = {'\0'};//receive the input set
    char rec2[100] = {'\0'};//the same as rec[100]
    char str[1+1] = {'\t', '\0'};//string for comparison

    printf("Please input the character set:");
    scanf("%[^\n]", rec);//'\n' as a separator

    for(int i = 0; i < strlen(rec); i++)
    {
        rec2[i] = rec[i];
    }

    char* p = strstr(rec, str);
    while(p)
    {
        char* pos = strstr(rec, str);//identify the position in
rec[100]
        char* pos2 = strstr(rec2, str);//identify the position in
rec2[100]

        for(int k = 0; k < 5; k++)
        {
            *(pos+k) = ' ';//substitute '\t' for five space
characters
        }
```

```
        while(*pos2)//put backward the subsequent character string
        {
            *(pos+5) = *(pos2+1);
            pos2 = pos2 + 1;
            pos = pos + 1;
        }

        p = strstr(rec, str);

        for(int i = 0; i < strlen(rec); i++)
        {
            rec2[i] = rec[i];//make sure the two are exactly the same
list
        }
    }

    printf("The output character set is:   %s", rec);
    return 0;
}
```

Test Case

- My program can deal with multiple '\t' situations and can distinguish '\t' from simply whitespace.

Characteristics and Side Effects

- I use `scanf("%[^\n]", rec);` to make sure that space is not counted as a separator. Only '\n' can separate the input stuff.
- I use a duplicate list `char rec2[100] = {'\0'};//the same as rec[100]` to make sure that while putting backward the subsequent char string, it will not change the char string itself for the robustness of the program.
- There are further extension space, in which the user can access a function that replaces whatever the user wants with whatever he wants.

Problem 3

Program 3

```c
/*----------------------------------------
Module name: program_3.c
Description: This is a display program that prompts the user for a
number and a base then displays that
number in requested base. Assume the set of bases is {dec, oct, hex}.
For this design, I also identify the possible error conditions.
Author: Bo Yue
Rev. 0 07 Aug 2019
----------------------------------------*/

#include <stdio.h>

//macro for three requested base display
#define decDisplay(num) printf("The number in requested base
is:%d\n", num)
#define octalDisplay(num) printf("The number in requested base
is:%o\n", num)
#define hexaDisplay(num) printf("The number in requested base
is:%x\n", num)

int main()
{
    //definition of receive variable
    int num = 0;
    char base[3+1] = {'\0'};

    //number input
    printf("Please input a number in decimal:");
    scanf("%d", &num);

    //requested base input
    printf("Please input your requested base:");
    scanf("%s", base);

    //four conditions
    if     ('d' == base[0] && 'e' == base[1] && 'c' == base[2] &&
'\0' == base[3]) decDisplay(num);
    else if('o' == base[0] && 'c' == base[1] && 't' == base[2] &&
'\0' == base[3]) octalDisplay(num);
    else if('h' == base[0] && 'e' == base[1] && 'x' == base[2] &&
'\0' == base[3]) hexaDisplay(num);
    else printf("Wrong base!!!\n");

    return 0;
}
```

Test Case

```
Please input a number in decimal:12
Please input your requested base:hex
The number in requested base is:c
```

```
Please input a number in decimal:15
Please input your requested base:oct
The number in requested base is:17
```

```
Please input a number in decimal:17
Please input your requested base:dec
The number in requested base is:17
```

```
Please input a number in decimal:17
Please input your requested base:hexa
Wrong base!!!
```

- The test case is as expected for base-changed number.

Characteristics and Side Effects

- The program use macro directives to simplify the main() body.
- Possible error conditions occur when *a non-integer is entered for the variable num* (a fractional number is entered or a character is entered, for example), and *the number entered exceeds the boundary of int type* .
- The first condition can be solved by applying `while(0 == scanf("%d", &num))` . The second has something to do the compiler itself.

Problem 4

Program 4

```c
/*----------------------------------------
Module name: program_4.c
Description: This is a display program that prompts the user for a
number and a base then displays that
number in requested base. Assume the set of bases is {dec, oct, hex}.
For this design, I also identify the possible error conditions.
Author: Bo Yue
Rev. 0 08 Aug 2019
----------------------------------------*/

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

//The function receives user input and return the result in a
character array
char* baseOperation(int num1,int num2,char* base,char op);

int result=0;
```

```
int len=0;
char sign= '\0';
char temp[50+1],result_array[50+1];

int main(){
    //create the working variables
    int num1 = 0;
    int num2 = 0;
    char base[4];
    base[3] = '\0';
    char op = '\0';

    printf("Please enter two positive decimal numbers, a base and an
operator:");

    //read user input and store them in local variables
    scanf("%d %d %c%c%c
%c",&num1,&num2,&base[0],&base[1],&base[2],&op);

    printf("The result of the operation is:%s",
baseOperation(num1,num2,base,op));
}

char* baseOperation(int num1,int num2,char* base,char op)
{
    //get the result of operation in decimal base
    if (op=='+')
        result=num1+num2;
    else if (op=='-')
        result=num1-num2;
    else if (op=='*')
        result=num1*num2;
    else if (op=='/')
        result=num1/num2;
    //if none of the operator is matched
    else
        return "Operator Error!!!";

    //get the sign of the result
    if (result<0){
        sign='-';
        result=-result;
    }
    else{
        sign = ' ';
    }
```

```
    //change the representation of the result in different bases
    //decimal base
    if (strcmp(base,"dec")==0){
        //get the reversed character array of the result
        while (result>=10){
            temp[len]=result%10+'0';
            len+=1;
            result/=10;
        }
        temp[len]=result+'0';
        len+=1;
        //first character represents the sign of the result
        result_array[0]=sign;
        //turn the reverse array back
        for (int i=0;i<len;++i){
            result_array[len-i]=temp[i];
        }
        //add the end-of-string note
        result_array[len+1]='\0';
        //return the result
        return result_array;

    }
    //octal base
    else if (strcmp(base,"oct")==0){
        //get the reversed character array of the result
        while (result>=8){
            temp[len]=result%8+'0';
            len+=1;
            result/=8;
        }
        temp[len]=result+'0';
        len+=1;
        //first character represents the sign of the result
        result_array[0]=sign;
        //turn the reverse array back
        for (int i=0;i<len;++i){
            result_array[len-i]=temp[i];
        }
        //add the end-of-string note
        result_array[len+1]='\0';
        //return the result
        return result_array;
    }
    //hexadecimal base
    else if (strcmp(base,"hex")==0){
        //get the reversed character array of the result
```

```
        while (result>=16){
            if (result%16<=9)
                temp[len]=result%16+'0';
            else
                temp[len]=result%16+'A'-10;
            len+=1;
            result/=16;
        }
        if (result<=9)
            temp[len]=result+'0';
        else
            temp[len]=result+'A'-10;
        len+=1;
        //first character represents the sign of the result
        result_array[0]=sign;
        //turn the reverse array back
        for (int i=0;i<len;++i){
            result_array[len-i]=temp[i];
        }
        //add the end-of-string note
        result_array[len+1]='\0';
        //return the result
        return result_array;
    }

    else
        return " Base Error!!!";
}
```

Test Case

```
Please enter two positive decimal numbers, a base and an operator:11 -9 dec *
The result of the operation is:-99
```

```
Please enter two positive decimal numbers, a base and an operator:12 99 oct -
The result of the operation is:-127
```

```
Please enter two positive decimal numbers, a base and an operator:80 8 hex /
The result of the operation is: A
```

```
Please enter two positive decimal numbers, a base and an operator:16 17 hea +
The result of the operation is: Base Error!!!
```

- The outcomes are as expected.

Characteristics and Side Effects

- Negative number is also taken into account.
- I do not use a format input and output of base. Instead, I use an array to save the input and the output.

- Possible error conditions are that: the input can not go beyond the boundary, and the base & operator characters are not as expected.

Problem 5

Program 5

```
![problem_5](C:\Users\Bobyue\Desktop\UW\Intro To C &
Microprocessors\homework3\problem_5.jpg)/*-------------------------
------------
Module name: Histogram.c
Description: Write a program Histogram that prompts the user to enter
a series of characters then
keeps count of the number of alphas (letter or digit), digits (0..9),
punctuation, and
whitespace (\t, \v, \f, \r, or \n) characters in the input data set.
When the user enters the
EOF character, print out the number of entries in each category.
Author: Bo Yue
Rev. 0 08 Aug 2019
---------------------------------------*/

#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>

#define CNT 1
int main()
{
    //declaration and initialization of variables
    int num_of_alpha = 0;
    int num_of_digit = 0;
    int num_of_punc  = 0;
    int num_of_space = 0;

    //variable for receiving input char
    char receive;
    //for error detection
    int result = 0;

    //input set
    printf("Please enter your words, with EOF at the end of the
input:");
```

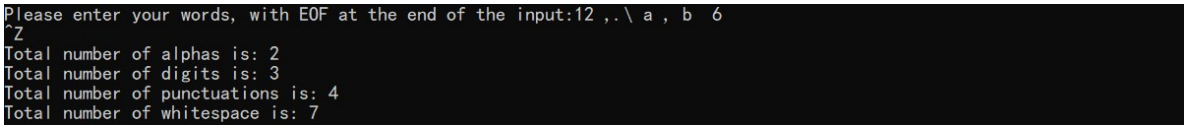
```
    while(1 == (result = scanf("%c", &receive)))
    {
        //count the number of entries in each category
        if(isalpha(receive)) {num_of_alpha++;}
        if(isdigit(receive)) {num_of_digit++;}
        if(ispunct(receive)) {num_of_punc++;}
        if(isspace(receive)) {num_of_space++;}
    }

    while (EOF != result) // combination of ctrl and z keys on a PC
or ctrl and d in Linux
    {
      printf("WARNING: Bad input after reading data!!!\n");
    }

    //output of number of entries in each category
    printf("Total number of alphas is: %d\n", num_of_alpha);
    printf("Total number of digits is: %d\n", num_of_digit);
    printf("Total number of punctuations is: %d\n", num_of_punc);
    printf("Total number of whitespace is: %d\n", num_of_space);

    return 0;
}
```

Test Case



- The outcomes is as expected. Since windows OS requires EOF to be at the beginning of the new line, the number of whitespace takes the character '\n' before the EOF into account.

Characteristics and Side Effects

- The program requires EOF at the end of the input. Otherwise, the program will loop.
- Only English characters are counted and no Chinese characters are allowed in the input session.

Statement

- I acknowledge that I code all the program myself.
- Signature: Bo Yue