PES 2019 Project 1 Readme

Utkarsh Dviwedi

The repository has been organized into project folders and program subfolders. There is a test folder for testing the git commit instructions. The Repository is public access. The description of the codes is given in the comments

Notes for running the code

Please enter a number on prompt for problem 2

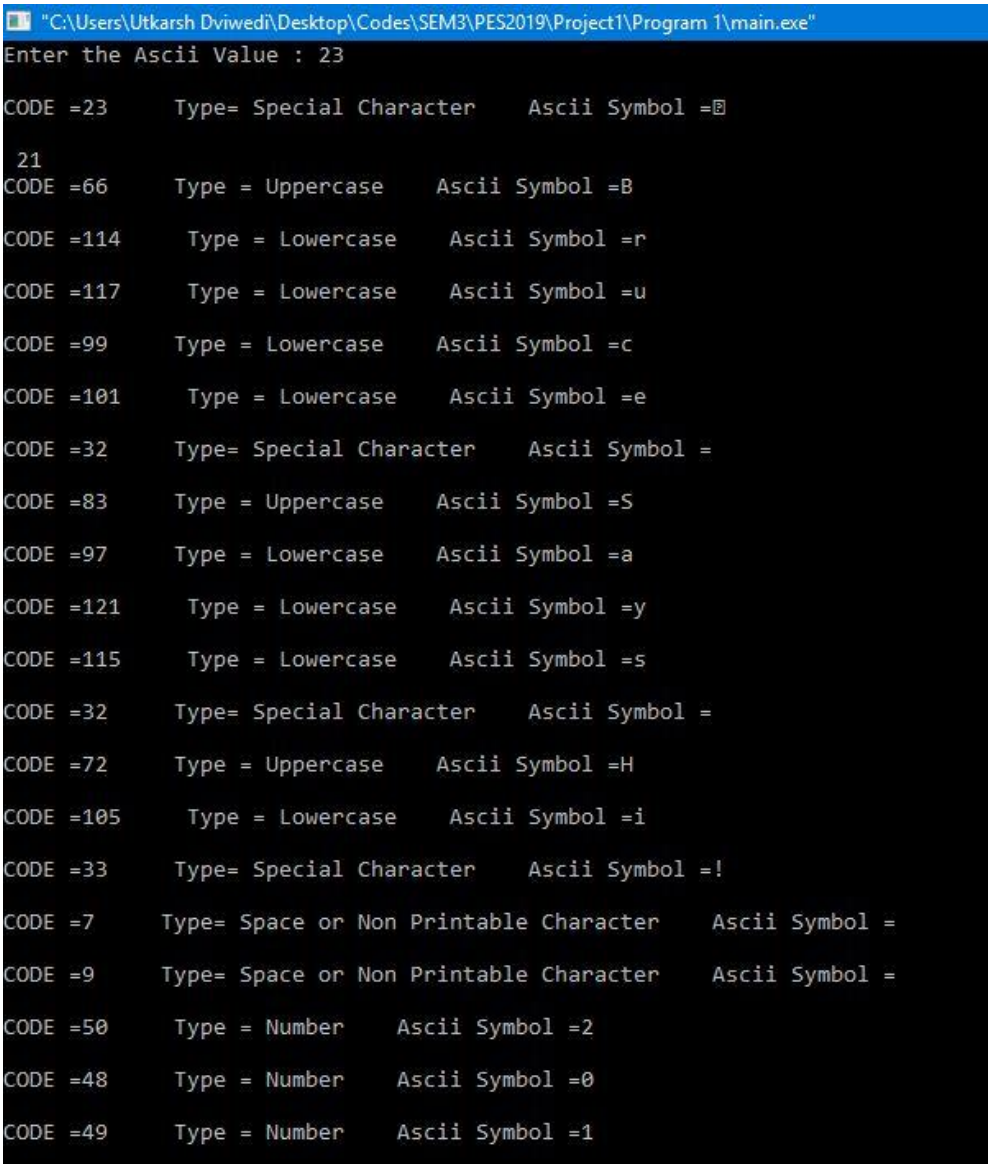
Program 2

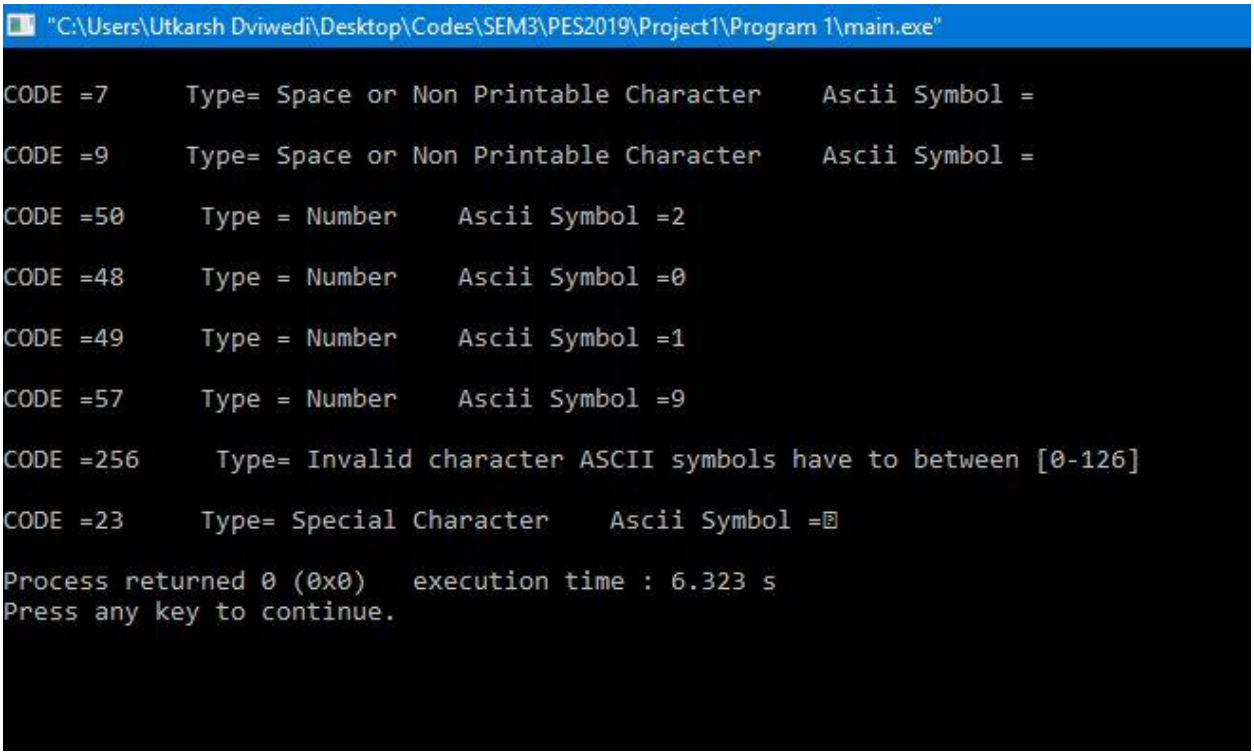
```
/*
Utkarsh Dviwedi 2019
Principles of Embedded Software Fall 2019
Project 1 Program 2

Question
Write a program that uses a logical expression that tests whether a given character code is a
lower case [97-122]
upper case [65-90]
digit [48-57]
white space (like null, backspace, space, tabs, etc.) [0-15,20]
or a special character (like ! or >) in ASCII.   [others less than 127]

References
https://stackoverflow.com/questions/6660145/convert-ascii-number-to-ascii-character-in-c
https://en.wikipedia.org/wiki/ASCII
*/

#include <stdio.h>
#include <stdlib.h>

//Function Prototype
void TypeDetector (int Ascii,char c_Ascii);


int main()
{
```

```
  int Ascii,i;
  char test;

  printf("Enter the Ascii Value : ");
  scanf("%d", &Ascii);
  char c_Ascii = Ascii;

  TypeDetector(Ascii,c_Ascii);

  int Ascii_Vals[] = {66,114,117,99,101,32,83,97,121,115,32,72,105,33,7,9,50,48,49,57,256};
  int Vals_count = ((sizeof(Ascii_Vals))/(sizeof(Ascii_Vals[0])));
  printf("\n %d",Vals_count);

  for (i=0;i<=Vals_count;i++)
  {
     c_Ascii = Ascii_Vals[i];
     TypeDetector(Ascii_Vals[i],c_Ascii);
  }


return 0;
}




//Function to Detect the Ascii symbol for an int
void TypeDetector (int Ascii,char c_Ascii)
```

```
{
   //For Lowercase Characters
    if (Ascii >= 97 && Ascii <= 122 )
  {
    printf("\nCODE =%d     Type = Lowercase    Ascii Symbol =%c \n",Ascii,c_Ascii);
  }

  //For digits
  else if (Ascii >= 48 && Ascii <= 57 )
  {
    printf("\nCODE =%d     Type = Number    Ascii Symbol =%c \n",Ascii,c_Ascii);
  }

  //For Uppercase Characters
  else if (Ascii >= 65 && Ascii <= 90 )
  {
    printf("\nCODE =%d     Type = Uppercase    Ascii Symbol =%c \n",Ascii,c_Ascii);
  }

  //For Invalid Characters
  else if (Ascii > 127  )
  {
    printf("\nCODE =%d     Type= Invalid character ASCII symbols have to between [0-126]   \n",Ascii);
  }

  else if ( (Ascii== 20) || ((Ascii <= 15 && Ascii >=0 )) )
  {
     printf("\nCODE =%d     Type= Space or Non Printable Character    Ascii Symbol =%c \n",Ascii,c_Ascii);
  }
```

```
  //For Special Characters
  else
  {
    printf("\nCODE =%d     Type= Special Character    Ascii Symbol =%c \n",Ascii,c_Ascii);
  }


}
```




"C:\Users\Utkarsh Dviwedi\Desktop\Codes\SEM3\PES2019\Project1\Program 1\main.exe"
CODE =7 Type= Space or Non Printable Character Ascii Symbol =
CODE =9 Type= Space or Non Printable Character Ascii Symbol =
CODE =50 Type = Number Ascii Symbol =2
CODE =48 Type = Number Ascii Symbol =0
CODE =49 Type = Number Ascii Symbol =1
CODE =57 Type = Number Ascii Symbol =9
CODE =256 Type= Invalid character ASCII symbols have to between [0-126]
CODE =23 Type= Special Character Ascii Symbol =
Process returned 0 (0x0) execution time : 6.323 s
Press any key to continue.

Program 3

/*

Utkarsh Dviwedi 2019

Principles of Embedded Software Fall 2019

Project 1 Program 3

Question

Given the starting integer value 0xCAFE, perform each of these operations in series,

that is, each operation should be performed on the result of the previous function.

1 Print the results of each function to the command line (to capture as ProgramThree.out).

2 Print the original input in hexadecimal

3 Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test – true/false)

4 Reverse the byte order, print the value in hexadecimal

5 Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test – true/false)

6 Rotate the value by four bits to the left, print the value in hexadecimal

7 Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test – true/false)

8 Rotate the value by eight bits to the right, print the value in hexadecimal

9 Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test – true/false)

References

https://codeforwin.org/2015/08/c-program-to-convert-hexadecimal-to-bin_tempary-number-system.html

https://www.tutorialspoint.com/cprogramming/c_return_arrays_from_function.htm

*/

```
#include<stdlib.h>
#include <stdio.h>
#include <string.h>    //For Strcat Functin
int bitsize =  15;

////////////////////////////////
//Function Prototypes//////////
////////////////////////////////

//Function to Rotate an array(bintemp) left(Dir 0) or right(Dir 1) by index rotate value
char *rotate(char bin_temp[],int rotate_val,int bitsize,int direction);

//Function to Reverse an Array.
char *reversal(char bin_temp[],int bitsize );

//Function to Convert 4 char bits to a Hex Number
char bit4_binary_to_hex(char i,char j,char k,char l);

//Function to Print the
void PrintHex(char bin_temp[]);

// Function to Check if three of the last four Bits are ON
void test3on(char bin[]);

// Function to Check if three of the last four Bits are ON
void test3on(char bin[]);

//Function to Convert a Hex Value to Binary
void HexToBin(char hex[],char *bin);
```

```
////////////////////////////////
//////////Main Function//////////
////////////////////////////////

int main()
{

   char bin[17];
   char * binptr;
   binptr =&bin;
   char hex[] = "CAFE";
   char *Rev;
   char *Rotated1;
   char *Rotated2;

   HexToBin(hex,binptr);

   printf("\n Hexademial number = %s\n", hex);
   printf("\n bin binary number = %s", bin);

   test3on(binptr);
   PrintHex(bin);

   printf("\n Assuming all operations to be cascaded");
   printf("\n\n ---------- Reversal of the bits---------------------------------------------- ");
   Rev = reversal(bin,bitsize);
   printf("\n the Binary value now is  %s",Rev);
```

```
   test3on(Rev);
   PrintHex(Rev);

   printf("\n\n ---------- Rotation 1 of the reversed bits by 4 bits to the left---------- ");
   Rotated1 = rotate(Rev,4,16,0);//Rotation by 4bits to the left
   test3on(Rotated1);
   PrintHex(Rotated1);


    printf("\n\n ----------  Rotation 2 of previously rotated array by 8 bits to the right------ ");
   Rotated2 = rotate(Rotated1,4,8,1);//Rotation by 8 bits to the Right
   test3on(Rotated2);
   PrintHex(Rotated2);

   printf("\n\n----------Program Completed----------\n\n");
   return 0;
}




////////////////////////////////////
//////////Function Definitions///////
////////////////////////////////////



//Function to Rotate an array(bintemp) left(Dir 0) or right(Dir 1) by index rotate value
char *rotate(char bin_temp[],int rotate_val,int bitsize,int direction)
```

```
{
   static char rotated[17];
   int i;
   rotated[16]='\0';

   //Left Rotation
   if (direction==0)
   {
      for (i=0;i<=bitsize-rotate_val;i++)
      {
         rotated[i]=bin_temp[i+rotate_val];
      }
      for (i=bitsize-rotate_val;i<bitsize;i++)
      {
         rotated[i]=bin_temp[i-(bitsize-rotate_val)];
      }
      printf("\n Original Number      %s",bin_temp);
      printf("\n Number after Rotation %s",rotated);
   }

   //Right Rotation
   if (direction==1)
   {
      rotate_val = bitsize - rotate_val;

      for (i=0;i<=bitsize-rotate_val;i++)
      {
         rotated[i]=bin_temp[i+rotate_val];
      }
```

```
        for (i=bitsize-rotate_val;i<bitsize;i++)
        {
           rotated[i]=bin_temp[i-(bitsize-rotate_val)];
        }
        printf("\n Original Number      %s",bin_temp);
        printf("\n Number after Rotation %s",rotated);

    }

return rotated;
}

//Function to Reverse an Array
char *reversal(char bin_temp[],int bitsize )
{
    static char reversed[17];
    int i;

    for (i=0;i<=bitsize;i++)
    {
        reversed[i] = bin_temp[bitsize-i];
    }
    reversed[16]='\0';

    return reversed;
}

//Function to Convert 4 char bits to a Hex Number
char bit4_binary_to_hex(char i,char j,char k,char l)
```

```
{
   if ( (i==48) && (j==48) && (k==48) && (l==48) )
   {return '0';}
   else if ( (i==48) && (j==48) && (k==48) && (l==49) )
   {return '1';}
   else if ( (i==48) && (j==48) && (k==49) && (l==48) )
   {return '2';}
   else if ( (i==48) && (j==48) && (k==49) && (l==49) )
   {return '3';}
   else if ( (i==48) && (j==49) && (k==48) && (l==48) )
   {return '4';}
   else if ( (i==48) && (j==49) && (k==48) && (l==49) )
   {return '5';}
   else if ( (i==48) && (j==49) && (k==49) && (l==48) )
   {return '6';}
   else if ( (i==48) && (j==49) && (k==49) && (l==49) )
   {return '7';}
   else if ( (i==49) && (j==48) && (k==48) && (l==48) )
   {return '8';}
   else if ( (i==49) && (j==48) && (k==48) && (l==49) )
   {return '9';}
   else if ( (i==49) && (j==48) && (k==49) && (l==48) )
   {return 'A';}
   else if ( (i==49) && (j==48) && (k==49) && (l==49) )
   {return 'B';}
   else if ( (i==49) && (j==49) && (k==48) && (l==48) )
   {return 'C';}
   else if ( (i==49) && (j==49) && (k==48) && (l==49) )
   {return 'D';}
```

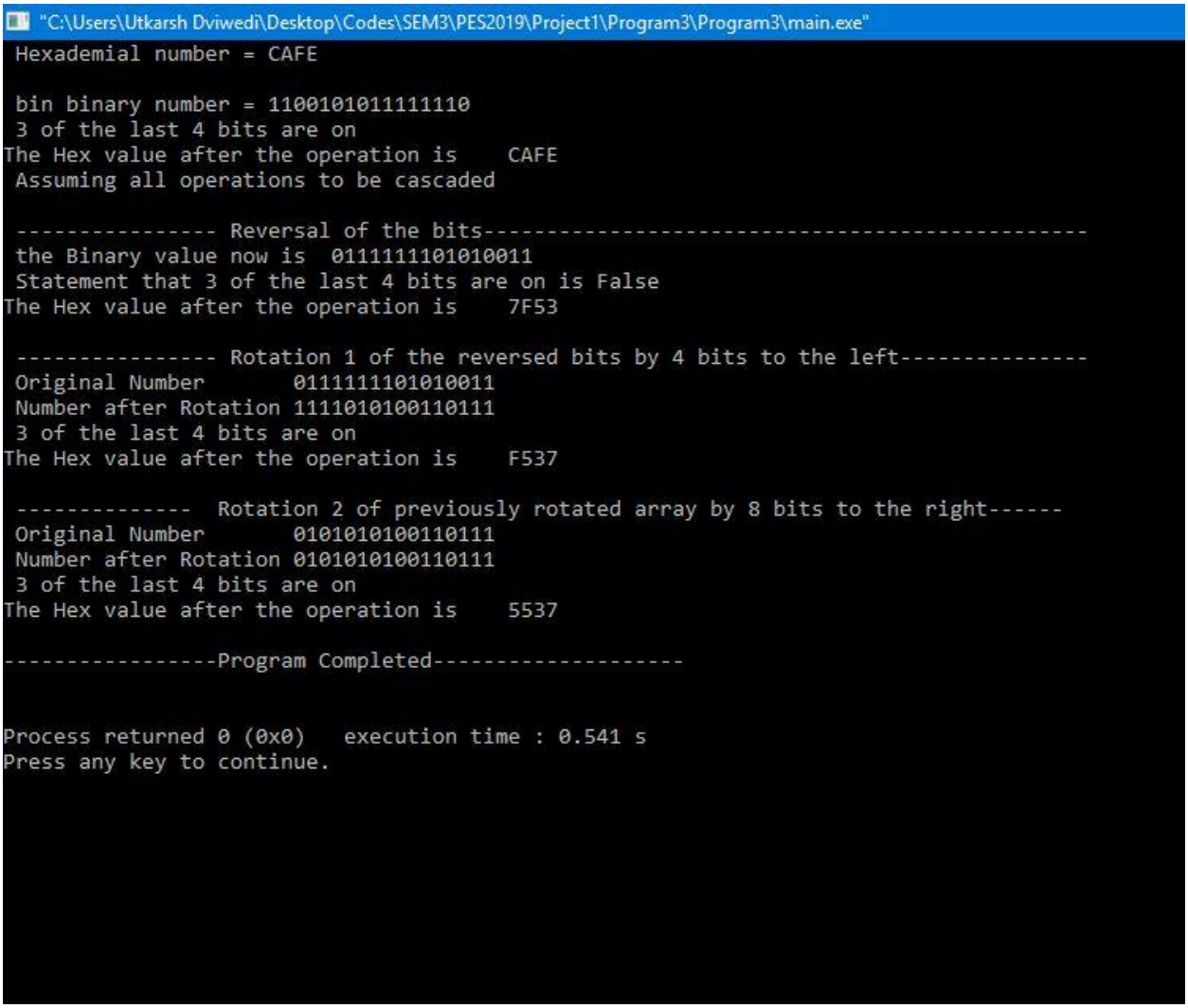
```
   else if ( (i==49) && (j==49) && (k==49) && (l==48) )
   {return 'E';}
   else if ( (i==49) && (j==49) && (k==49) && (l==49) )
   {return 'F';}
   else
   {return 'Z';}
}

//Function to Print the
void PrintHex(char bin_temp[])
{
 int i;
 printf("\nThe Hex value after the operation is    ");
 for (i=0;i<=3;i++)
 {

printf("%c",bit4_binary_to_hex((int)bin_temp[i*4],(int)bin_temp[(i*4)+1],(int)bin_temp[(i*4)+2],(int)bin
_temp[(i*4)+3]));
 }
}

// Function to Check if three of the last four Bits are ON
void test3on(char bin[])
{
   int count = 0;
   int i = 0;
```

```
   for (i = 12; i < 16; i++)
   {
      if (bin[i]=='1')
      {
         count++;
      }
   }

   if (count == 3 )
   {
      printf("\n 3 of the last 4 bits are on");
   }
   else
   {
      printf("\n Statement that 3 of the last 4 bits are on is False ");
   }

}


//Function to Convert a Hex Value to Binary
void HexToBin(char hex[],char *bin)
{
   char bin_temp[65] = "";
   //char bin[17];
   int i = 0;

   /* Extract first digit and find bin_temporary of each hex digit */
   for(i=0; hex[i]!='\0'; i++)
```

```
{
   switch(hex[i])
   {
      case '0':
         strcat(bin_temp, "0000");
         break;
      case '1':
         strcat(bin_temp, "0001");
         break;
      case '2':
         strcat(bin_temp, "0010");
         break;
      case '3':
         strcat(bin_temp, "0011");
         break;
      case '4':
         strcat(bin_temp, "0100");
         break;
      case '5':
         strcat(bin_temp, "0101");
         break;
      case '6':
         strcat(bin_temp, "0110");
         break;
      case '7':
         strcat(bin_temp, "0111");
         break;
      case '8':
         strcat(bin_temp, "1000");
```

```
    break;
case '9':
    strcat(bin_temp, "1001");
    break;
case 'a':
case 'A':
    strcat(bin_temp, "1010");
    break;
case 'b':
case 'B':
    strcat(bin_temp, "1011");
    break;
case 'c':
case 'C':
    strcat(bin_temp, "1100");
    break;
case 'd':
case 'D':
    strcat(bin_temp, "1101");
    break;
case 'e':
case 'E':
    strcat(bin_temp, "1110");
    break;
case 'f':
case 'F':
    strcat(bin_temp, "1111");
    break;
```

```
    }
  }



  printf("\n binary number = %s", bin_temp);



  //Assign the binary value to
  for (i=0;i<=15;i++)
  {
    bin[i]=bin_temp[i];
  }
  bin[16]='\0';
  printf("\n bin binary number = %s", bin);

}
```


"C:\Users\Utkarsh Dviwedi\Desktop\Codes\SEM3\PES2019\Project1\Program3\Program3\main.exe"
Hexademial number = CAFE
bin binary number = 1100101011111110
3 of the last 4 bits are on
The Hex value after the operation is CAFE
Assuming all operations to be cascaded
---------- Reversal of the bits----------
the Binary value now is 0111111101010011
Statement that 3 of the last 4 bits are on is False
The Hex value after the operation is 7F53
---------- Rotation 1 of the reversed bits by 4 bits to the left----------
Original Number 0111111101010011
Number after Rotation 1111010100110111
3 of the last 4 bits are on
The Hex value after the operation is F537
---------- Rotation 2 of previously rotated array by 8 bits to the right------
Original Number 0101010100110111
Number after Rotation 0101010100110111
3 of the last 4 bits are on
The Hex value after the operation is 5537
----------Program Completed----------
Process returned 0 (0x0) execution time : 0.541 s
Press any key to continue.