1DT301 lab6

Songho Lee Sarpreet Singh Buttar

October 13, 2016

1 Introduction

This report provides the solutions for the sixth laboratory of the course 1DT301, which is focusing on utilising of the Cybertech display and serial communication of STK600, and writing programmes in C language which we barely learnt the basic syntax.

For all tasks it is configured to run on the CPU oscillation rate of 1.8432MHz, and we are using UBRR baud rate as 2400bps.

2 Task 1

Task one programme writes a character on the CyberTech Display. Our programme writes 'A' on the display. Code and flowchart follows.

```
#include <avr/io.h>
#include <stdio.h>
#define BAUD 2400
#define UBRR_VAL 47
//#define BAUDRATE ((F_CPU)/(BAUD*16UL)-1)


void uart_int(void);
void uart_trans(unsigned char data);

int main(void){

  uart_int();

  char* temp = "\rAO0001A";

  int i;
  int checksum=0;

  for(i=0;i<8;i++){
    checksum+=temp[i];
  }

  checksum%=256;

  char towrite[10];

  sprintf(towrite,"%s%02X\n",temp,checksum);

  for(i=0;i <11;i++){
    uart_trans(towrite[i]);
  }

  temp = "\rZD0013C\n";
  for(i=0;i<9;i++)
  {
    uart_trans(temp[i]);
  }

```

```
40   return 0;
41 }
42
43 void uart_int(void){
44   UBRR1L = UBRR_VAL;
45   UCSR1B = (1<<TXEN1) | (1<<RXEN1);
46
47 }
48 void uart_trans(unsigned char data){
49   while (!( UCSR1A & (1<<UDRE1)));                // wait while register is free
50
51     UDR1 = data;                                   // load data in the register
52 }
```

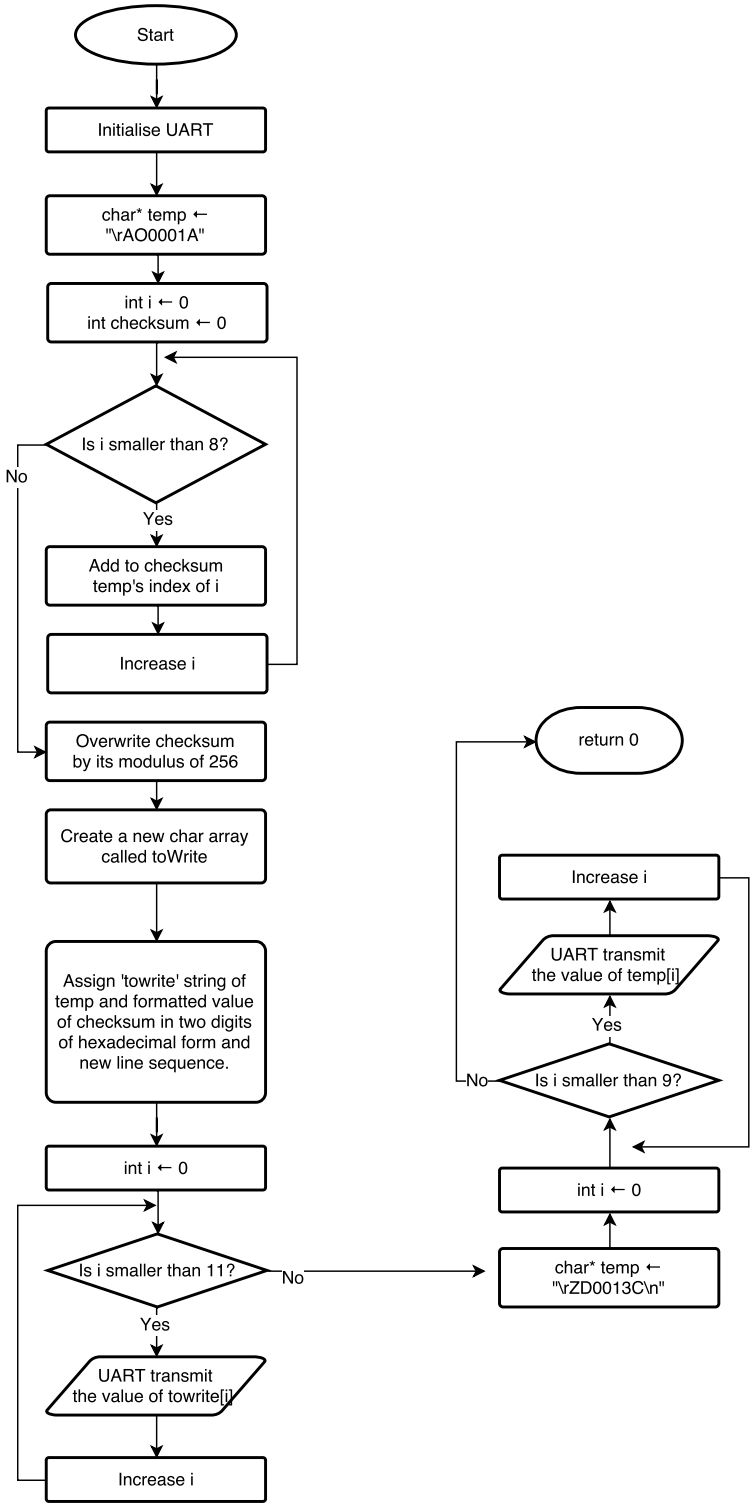

Figure 1: Flowchart of task 1

3 Task 2

Our programme writes Hello in Korean in first line, Hello in Punjab language in second line and prints third line that it is done. Code and flowchart follows.

```
#include <avr/io.h>
#include <stdio.h>
#define BAUD 2400
#define UBRR_VAL 47
#define N 50


void uart_int(void);
void uart_trans(unsigned char data);
void printout(char* inchar, int length);

int main(void){

  uart_int();

  char* temp = "\rAO0001An nyeong ha se yo       Sat Sri Akaal";
  printout(temp,45);

  temp = "\rBO0001Done!";
  printout(temp,13);

  temp = "\rZD001";
  printout(temp,6);

  return 0;
}

void uart_int(void){
  UBRR1L = UBRR_VAL;
  UCSR1B = (1<<TXEN1) | (1<<RXEN1);

}
void uart_trans(unsigned char data){
  while (!( UCSR1A & (1<<UDRE1)));                // wait while register is free

    UDR1 = data;                                   // load data in the register
}
void printout(char* inchar, int length){
  int i,checksum;

  checksum=0;
  for(i=0;i<length;i++){
    checksum+=inchar[i];
  }

  checksum%=256;

  char towrite[N];

  sprintf(towrite,"%s%02X\n",inchar,checksum);

  for(i=0;i<length+3;i++)
  {
    uart_trans(towrite[i]);
  }
}
```

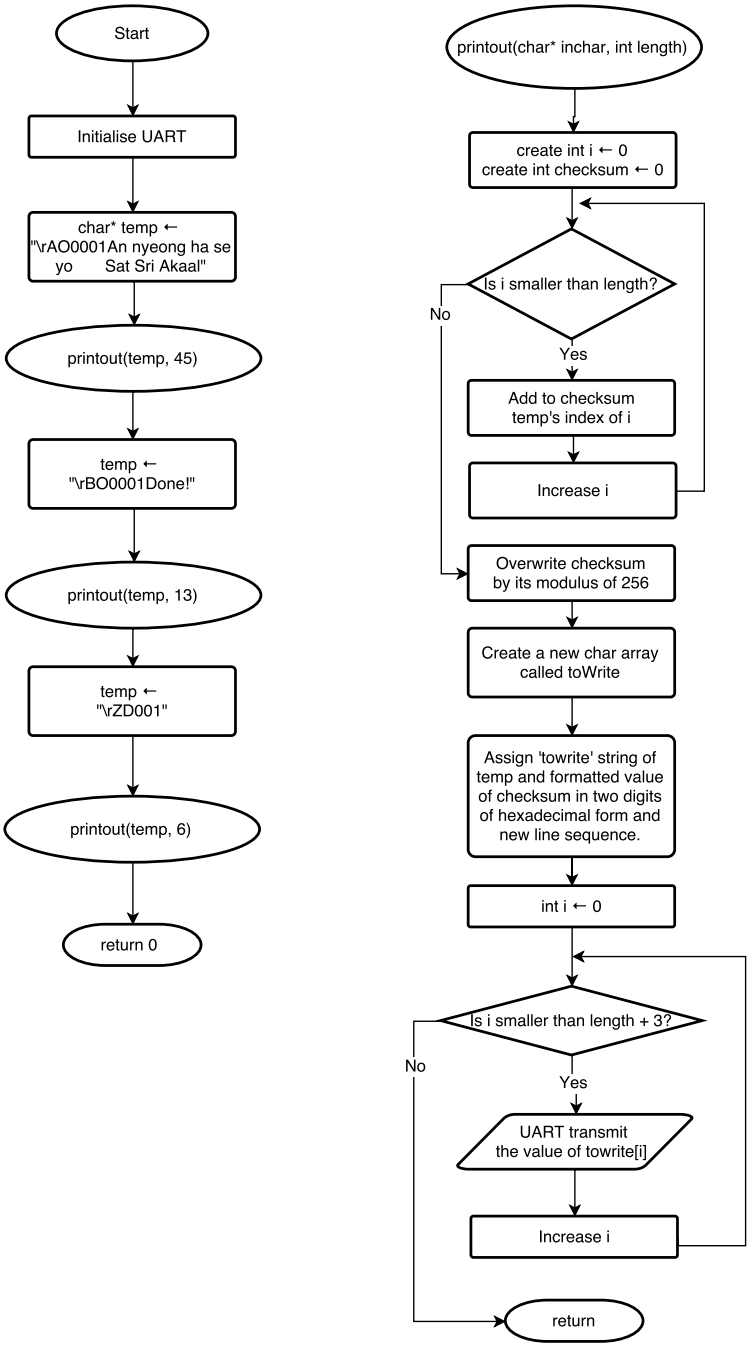

Figure 2: Flowchart of task 2

4 Task 3

Task three displays greetings to Benjamin, asks if he thinks this programme works and says we think it is works and ends up with printing out name. It scrolls over and repeats infinitely. The code and chart is presented below.

```
#include <avr/io.h>

#define F_CPU 1843200UL
#include <util/delay.h>

#include <stdio.h>
#include <string.h>
#define BAUD 2400
#define UBRR_VAL 47
#define N 150
//#define CLOCK 1843200



void uart_int(void);
void uart_trans(unsigned char data);
void printout(char memory,char* inchar);
char* customconcat(char* inchar1, char* inchar2);
void showondisplay();

int main(void){

  uart_int();

  char* strings[4]={"Hello Benjamin","Do you think this works?", "we think so.", "
    Songho, sarpreet"};
  char concatanated[N];

  int i=0;

  while(1)
  {
    strcpy(concatanated,customconcat(strings[i],strings[(i+1)%4]));
    printout('A',concatanated);
    //printout('A',strings[0]);
    printout('B',strings[(i+2)%4]);
    showondisplay();
    i++;
    if(i==4)
      i=0;
  }
  return 0;
}

void uart_int(void){
  UBRR1L = UBRR_VAL;
  UCSR1B = (1<<TXEN1) | (1<<RXEN1);

}
void uart_trans(unsigned char data){
  while (!( UCSR1A & (1<<UDRE1)));                // wait while register is free

    UDR1 = data;                                   // load data in the register
}
void printout(char memory,char* inchar){

  int i,length,checksum;

  char temp[N] = "\rAO0001";
  temp[1]=memory;

  strcat(temp,inchar);
  length = strlen(temp);
```

```
  checksum=0;

  for(i=0;i<length;i++){
    checksum+=temp[i];
  }
  checksum %=256;

  char towrite[N];
  sprintf(towrite,"%s%02X\n",temp,checksum);
  length = strlen(towrite);
  for(i=0;i<length;i++)
  {
    uart_trans(towrite[i]);
  }

}
void showondisplay(){
  int i;
  char* temp = "\rZD0013C\n";
  for(i=0;i<9;i++)
  {
    uart_trans(temp[i]);
  }
  _delay_ms(5000);
}
char* customconcat(char* inchar1, char* inchar2){

  char toreturn[N];

  strcpy(toreturn,inchar1);
  if(strlen(toreturn) < 24)
  {
    int i;

    for(i=strlen(toreturn);i<24;i++)
    {
      strcat(toreturn, " ");
    }
  }

  strcat(toreturn, inchar2);
  return toreturn;
}
```

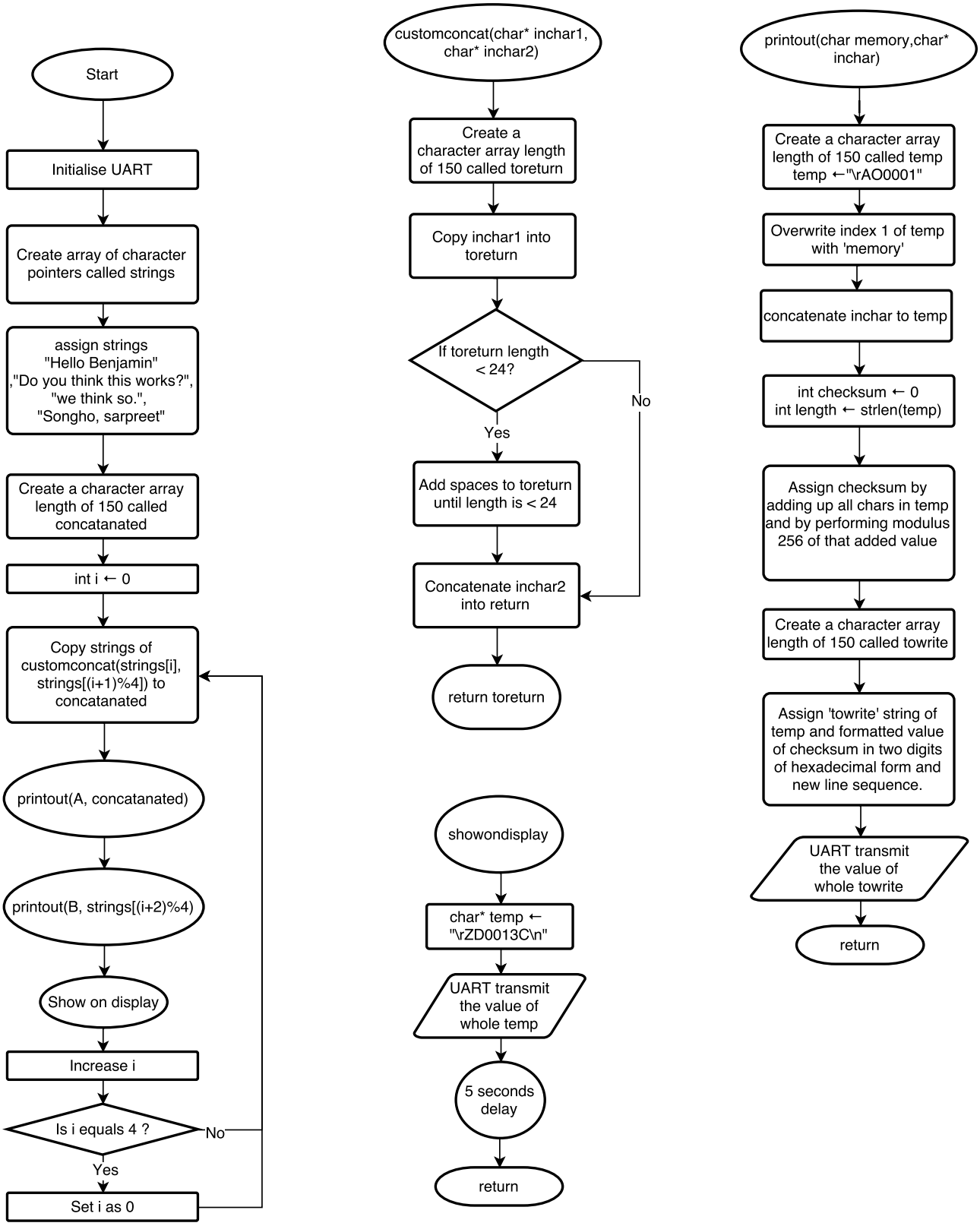

Figure 3: Flowchart of task 3

5 Task 4

Task 4 modified task three to receive inputs from PuTTY. Return sequence is '/'. and change line to certine line sequence is '>'. For example if someone enters '>2' it writes second line of the display. The code and flowchart is presented below.

```
#include <avr/io.h>

#define F_CPU 1843200UL
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>
#define BAUD 2400
#define UBRR_VAL 47
#define N 150
//#define CLOCK 1843200



void uart_int(void);
void uart_trans(unsigned char data);
void printout(char memory,char* inchar);
char* customconcat(char* inchar1, char* inchar2);
char USARTReadChar();
void showondisplay();

int main(void){

  uart_int();

  char inchars[4][N],inchar,memory;
  char tmp[N];
  int currentline=0;
  int lengths[4]={0,0,0,0};

  memory='A';
  printout('A',"Enter chars on PuTTY>");
  printout('B'," ");
  showondisplay();

  while(1)
  {
    inchar=USARTReadChar();

    inchars[currentline][lengths[currentline]]=inchar;
    lengths[currentline]++;


    //Line change sequence.
    if(inchar=='/' || inchar=='>'){
      if(inchar == '>'){
        inchar=USARTReadChar();
        currentline = (int) (inchar-49) % 3;
      }
      else
        currentline = (currentline+1) % 3;

      if(lengths[currentline]!=0)
      {
        memset(inchars[currentline],0,strlen(inchars[currentline]));
        lengths[currentline]=0;
      }
    }


    if(currentline!=2){
      memory = 'A';
      strcpy(tmp,customconcat(inchars[0],inchars[1]));
```

```
64        printout(memory,tmp);
65        memset(tmp,0,strlen(tmp));
66
67      }
68      else{
69        memory = 'B';
70        //strcpy(tmp, inchars[currentline]);
71        printout(memory,inchars[currentline]);
72
73      }
74      showondisplay();
75      /*
76      printout(memory,tmp);
77      showondisplay();
78      memset(tmp,0,strlen(tmp));
79      */
80    }
81
82    return 0;
83  }
84
85  void uart_int(void){
86    UBRR1L = UBRR_VAL;
87    UCSR1B = (1<<TXEN1) | (1<<RXEN1);
88
89  }
90  void uart_trans(unsigned char data){
91    while (!( UCSR1A & (1<<UDRE1)));                  // wait while register is free
92      UDR1 = data;                                     // load data in the register
93  }
94
95  void printout(char memory,char* inchar){
96
97    int i,length,checksum;
98
99    char temp[N] = "\rAO0001";
100   temp[1]=memory;
101
102   strcat(temp,inchar);
103   length = strlen(temp);
104   checksum=0;
105
106   for(i=0;i<length;i++){
107     checksum+=temp[i];
108   }
109   checksum %=256;
110
111   char towrite[N];
112   sprintf(towrite,"%s%02X\n",temp,checksum);
113   length = strlen(towrite);
114   for(i=0;i<length;i++)
115   {
116     uart_trans(towrite[i]);
117   }
118
119 }
120
121 char USARTReadChar(){
122
123    while(!(UCSR1A & (1<<RXC1)));
124
125    return UDR1;
126
127 }
128 void showondisplay(){
129   int i;
130   char* temp = "\rZD0013C\n";
131   for(i=0;i<9;i++)
132   {
133     uart_trans(temp[i]);
```

```
134   }
135
136 }
137
138 char* customconcat(char* inchar1, char* inchar2){
139
140   char toreturn[N];
141
142   strcpy(toreturn,inchar1);
143   if(strlen(toreturn) < 24)
144   {
145     int i;
146
147     for(i=strlen(toreturn);i<24;i++)
148     {
149       strcat(toreturn, " ");
150     }
151   }
152   strcat(toreturn, inchar2);
153
154   return toreturn;
155 }
```

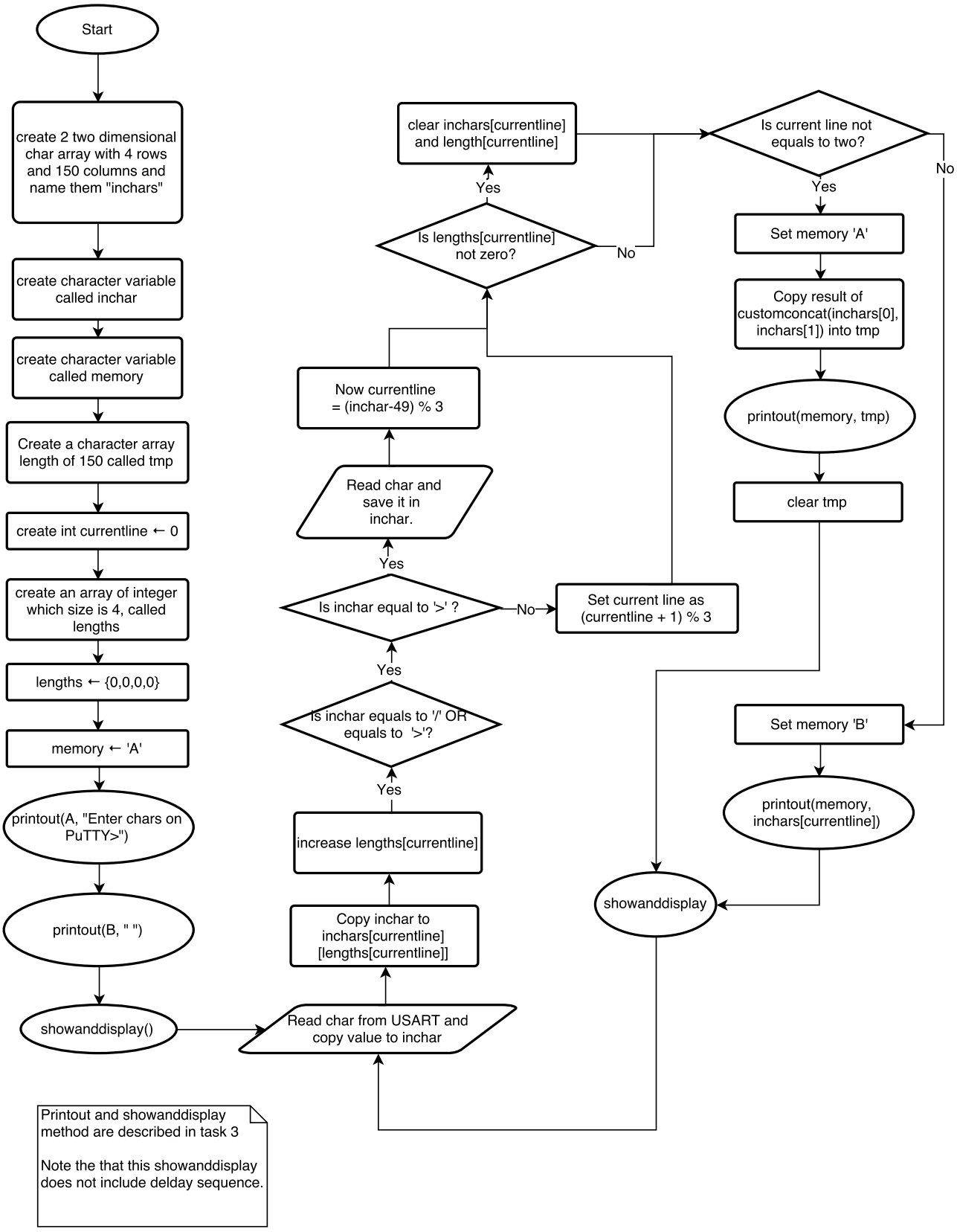

Figure 4: Flowchart of task 4