```
// Written by: HENG LOW WEE
// Student ID: U096901R
// Module: CS2104
// Problem Set 1

// Problem 1
#include <stdio.h>
#include <string.h>

int eax, ebx, ecx, edx, esi, edi;
// esi is the pointer to the string
// eax is the output
unsigned char M[10000];
void exec();

int main() {

    // These values are here for testing
    // let's say ebx is the length of string
    // start address is esi, end address is ebx-1
    esi = 1234;
    char a[] = {'9','9','9'};
    ebx = 3;
    memcpy(&M[esi], a, ebx);
    //

    exec();
    eax += 1; // just to make sure eax captured an int
    printf("When testing, we expect eax=999\n");
    printf("eax+1 = %d\n", eax);
    getchar();
}

void exec() {

    // ASCII code for digits 0-9 is 48-57
    edi = esi; // edi is used as the index
    edi += ebx; // add length to go to the last character
    edi -= 1; // start from the last character
    ecx = 1; // multiplier

    loop_start:

    if ( edi >= esi ) goto then_branch;
    goto loop_end;
```

```
    then_branch:
    edx = (int)M[edi]; // get the character's ASCII number
    edx -= 48; // and we have get the numeric value of the character
    edx *= ecx; // multiply according to its place. e.g. tenth, hundredth
    eax += edx; // add this number
    ecx *= 10; // increase its place
    edi -= 1; // decrement the index
    goto loop_start;

    loop_end:{}
}
```

```
// Written by: HENG LOW WEE
// Student ID: U096901R
// Module: CS2104
// Problem Set 1

// Problem 2
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

const char *alpha = "_abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
const char *digits = "0123456789";
const char *delim = ";";
const char *ops = "+-*/";
const char *equal = "=";
typedef enum {q0=0, q1, q2, q3, q4, q5} stateType;

typedef struct {
    char name[256];
    int value;
    } Variable;

Variable v_list[100];
int noOfV;

void print_state(stateType state) {
    switch (state) {
        case 0: printf("q0"); break;
```

```
        case 1: printf("q1"); break;
        case 2: printf("q2"); break;
        case 3: printf("q3"); break;
        case 4: printf("q4"); break;
        case 5: printf("q5"); break;
    }
}

int existIn(const char *arr, char c) {
    int length = strlen(arr);
    int i=0;
    for (i=0; i<length; i++) {
        if (arr[i] == c) {
            return 1;
        }
    }
    return 0;
}

int checkExist(char var[]) {
    int i=0;
    Variable v;
    for (i=0; i<noOfV; i++) {
        v = v_list[i];
        //printf("<v.name=%s vs. %s> ",v.name,var);
        if (!strcmp(v.name,var)) {
            return 1;
        }
    }
    return 0;
}

void addToList(char var[]) {
    Variable v;
    strcpy(v.name, var);
    v.value = 0;
    v_list[noOfV] = v;
    noOfV+=1;
}

void computeVV(char lvar[], char rvar[], char op[]) {
    Variable lv, rv, temp;
    int i=0;
    int left,right;
    int valLeft, valRight;
```

```
    for (i=0; i<noOfV; i++) {
        temp = v_list[i];
        if (!strcmp(temp.name, lvar)) {
            left = i;
            valLeft = temp.value;
        }
        else if (!strcmp(temp.name, rvar)) {
            right = i;
            valRight = temp.value;
        }
    }
    int result=0;
    if (!strcmp(op, "+")) result=valLeft+valRight;
    else if (!strcmp(op, "-")) result=valLeft-valRight;
    else if (!strcmp(op, "*")) result=valLeft*valRight;
    else if (!strcmp(op, "/")) result=valLeft/valRight;

    v_list[left].value = result;
}

void computeVI(char lvar[], char value[], char op[]) {
    int i=0;
    int left,right;
    int valLeft, valRight;
    Variable temp;
    for (i=0; i<noOfV; i++) {
        temp = v_list[i];
        if (!strcmp(temp.name, lvar)) {
            //printf("found\n");
            left = i;
            valLeft = temp.value;
        }
    }
    int num = atoi(value);
    int result=0;
    if (!strcmp(op, "+")) result=valLeft+num;
    else if (!strcmp(op, "-")) result=valLeft-num;
    else if (!strcmp(op, "*")) result=valLeft*num;
    else if (!strcmp(op, "/")) result=valLeft/num;

    v_list[left].value = result;
}

int main() {
    noOfV=0;
```

```
    char c;
    char lname[256];
    int lname_c=0;
    char rname[256];
    int rname_c=0;
    char value[256];
    int value_c=0;
    char op[2];
    int op_c=0;
    stateType state;
    state = q0;
    printf("Enter your input:\n");
    while ((c = getchar()) != '\n') {
        //printf("delta(");
        //print_state(state);
        //printf(", %c) = ", c);

        switch(state) {
            case q0:
            if (existIn(alpha, c)) {
                state = q1;
                lname[lname_c] = c;
                lname_c++;
            }
            else if (existIn(digits, c)) {
                printf("Explode!\n");
                exit(0);
            }
            break;

            case q1:
            if (existIn(alpha,c)) {
                state = q1;
                lname[lname_c] = c;
                lname_c++;
            }
            else if (existIn(digits, c)) {
                state = q1;
                lname[lname_c] = c;
                lname_c++;
            }
            else if (existIn(ops, c)) {
                state = q2;
                op[op_c] = c;
                op_c++;
```

```
            }
            else {
                printf("Explode!\n");
                exit(0);
            }
            break;

            case q2:
            // if we reach this state, lname is already determined
            // check whether it already exists
            lname[lname_c] = '\0';
            op[op_c] = '\0';
            if (!checkExist(lname)) {
                //printf("<lname=%s don't exist.> ", lname);
                addToList(lname);
            }

            if (existIn(equal, c)) {
                state = q3;
            }
            else {
                printf("Explode!\n");
                exit(0);
            }
            break;

            case q3:
            if (existIn(alpha, c)) {
                state = q4;
                rname[rname_c] = c;
                rname_c++;
            }
            else if (existIn(digits, c)) {
                state = q5;
                value[value_c] = c;
                value_c++;
            }
            break;

            case q4:
            if (existIn(alpha, c)) {
                state = q4;
                rname[rname_c] = c;
                rname_c++;
            }
```

```
            else if (existIn(digits, c)) {
                state = q4;
                rname[rname_c] = c;
                rname_c++;
            }
            else if (existIn(delim, c)) {
                state = q0;
                rname[rname_c] = '\0';
                // first check whether rname exist
                if (!checkExist(rname)) {
                    //printf("<rname=%s don't exist.> ", rname);
                    addToList(rname);
                }
                // carry out computation
                computeVV(lname, rname, op);

                // computation done, clear all
                lname[lname_c] = '\0';
                rname[rname_c] = '\0';
                value[value_c] = '\0';
                op[op_c] = '\0';
                lname_c = rname_c = value_c = op_c = 0;
            }
            break;

            case q5:
            if (existIn(digits, c)) {
                state = q5;
                value[value_c] = c;
                value_c++;
            }
            else if (existIn(delim, c)) {
                state = q0;
                value[value_c] = '\0';

                // carry out computation
                computeVI(lname, value, op);

                // computation done, clear all
                // computation done, clear all
                lname[lname_c] = '\0';
                rname[rname_c] = '\0';
                value[value_c] = '\0';
                op[op_c] = '\0';
                lname_c = rname_c = value_c = op_c = 0;
```

```
            }
            else if (existIn(alpha, c)) {
                printf("Explode!\n");
                exit(0);
            }
            break;
        }
        //print_state(state);
        //printf("\n");
    }
    printf("####################\n");
    printf("No. of variables = %d\n", noOfV);
    //print_state(state);
    int output=0;
    Variable out;
    for (output=0; output<noOfV; output++) {
        out = v_list[output];
        printf("Varible %s = %d\n", out.name, out.value);
    }
}
```

```
// Written by: HENG LOW WEE
// Student ID: U096901R
// Module: CS2104
// Problem Set 1

// Problem 3
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

int eax, ebx, ecx, edx, esi, edi;
unsigned char M[10000];
void exec();

int main() {
    int i=0;
    int total=0;
    int val;
```

```
    int num[] = {12, 34, 56, 128, 9};
    int ptr[] = {16, 56, 80, 64, 8, 0};

    // compute the expected total value
    for (i=0; i<5; i++) {
        val = num[i];
        total+=val;
    }
    ptr[5] = 0; // set last ptr to 0, which represents null pointer

    esi = ptr[0];
    int index = esi;
    for (i=0; i<5; i++) {
        printf("num=%d, ptr=%d\n", num[i], ptr[i+1]);
        *(int*)&M[index] = num[i];
        index+=4;
        *(int*)&M[index] = ptr[i+1];
        index = ptr[i+1];
    }
    printf("##################\n");

    exec();
    printf("expected total = %d\n", total);
    printf("eax = %d\n", eax);
    getchar();
}

void exec() {
    eax = 0;
    edi = esi; // edi is used to store the pointer

    loop_start:
    // get the value
    ebx = *(int*)&M[edi];
    // get the ptr
    ecx = *(int*)&M[edi+4]; // ecx is a temp for pointer
    edi = ecx;
    eax += ebx;

    if (edi) goto loop_start;
}
```