```c
#include <stdio.h>


double eval(double p[], double x, int n);
double add(double f[], double g[], double h[], int n);

int main(void) {

        int n;
        printf("ë\213¤í\225-ì\213\235ì\235\230 ì°¨ì\210\230 n: ");
        scanf("%d", &n);
        double p[101];  // ìµ\234ë\214\200 100ì°¨ê¹\214ì§\200 ê°\200ë\212¥í\225\230ë\217\204ë¡\235


        printf("xê°\222: ");
        double x;
        scanf("%lf", &x);


        printf("a0 ~ a%d ì\236\205ë ¥: ", n);
        for (int i = 0; i <= n; i++) {
                scanf("%lf", &p[i]);
        }


        double result = eval(p, x, n);
        printf("p(%.0f) = %.0f\n", x, result);



        printf("----------------------\n");

    // ë\221\220 ë\213¤í\225-ì\213\235ì\235\230 ë\215§ì\205\210 í\205\214ì\212¤í\212¸
        printf("ë\213¤í\225-ì\213\235ì\235\230 ì°¨ì\210\230 n: ");
        scanf("%d", &n);


        double g[101], h[101], f[101];


        printf("g(x) a0 ~ a%d ì\236\205ë ¥: ", n);
        for (int i = 0; i <= n; i++) {
                scanf("%lf", &g[i]);
        }


        printf("h(x) b0 ~ b%d ì\236\205ë ¥: ", n);
        for (int i = 0; i <= n; i++) {
                scanf("%lf", &h[i]);
        }


        add(f, g, h, n);

        // ê²°ê³¼ ì¶\234ë ¥
        printf("g(x) = %.0f", g[0]); //ì\206\214ì\210\230ì \220 ì\225\204ë\236\230ë¥¼ ê¹\224ë\201\224í\225\230ê²\214 ì \234ê±°í\225\230ê¸° ì\234\204í\225´ %.0fë¥¼ ì\202¬ì\232©í\225\234ë\213¤
        for (int i = 1; i <= n; i++) {
                printf(" + %.0fx^%d", g[i], i);
        }
        printf("\n");


        printf("h(x) = %.0f", h[0]);
        for (int i = 1; i <= n; i++) {
                printf(" + %.0fx^%d", h[i], i);
        }
        printf("\n");

        printf("f(x) = %.0f", f[0]);
        for (int i = 1; i <= n; i++) {
                printf(" + %.0fx^%d", f[i], i);
        }
        printf("\n");

        return 0;
}



double eval(double p[], double x, int n)
{
    double result = p[0]; // ì´\210ê¸°ê°\222ì\235\200 a0ë¡\234 ì¤\200ë\213¤ ì\235´ë\212\224 ì\203\201ì\210\230í\225-ì\235´ë\213¤
    double temp = x; //ì\236\204ì\213\234 ê³\200ì\210\230 tempì\227\220ë\212\224 ì\235¼ë\213¨ xë¥¼ ë\204£ì\226´ë\221\224ë\213¤
    for (int i = 1; i <= n; i++) {
        result += p[i] * temp; //i=1ì\235¼ë\225\214 ì\232°ì\204  a0xë¶\200í\204° ì\213\234ì\236\221í\225\230ì\227¬, iê°\200 ì¦\235ê°\200í\225¨ì\227\220 ë\224°ë\235¼ xì\235\230 ì°¨ì\210\230ë\217\204 í\225\230ë\202\230ì\224© ì¦\235ê°\200í\225\230ê²\214 ë§\214ë\223 ë\213¤
        temp *= x; // tempì\227\220 x^i ê°\222ì\235\204 ë\210\204ì \201
    }
    return result;
}


double add(double f[], double g[], double h[], int n)
{
    for (int i = 0; i <= n; i++) {
        f[i] = g[i] + h[i];
    }
    return f[n];
}
```

```c
#include <stdio.h>

int get_length(char *str);
int is_alpha(char c);
char to_lower(char c);
int check_palindrome_array(char str[]);
int check_palindrome_pointer(char *str);




int main(void)
{

        char str[100];//ë¬¸ì\236\220ì\227´ ê¸¸ì\235´ ìµ\234ë\214\200
100ì\234¼ë¡\234 ì\204¤ì \225í\225\234ë\213¤
        printf("Input: ");

        int i = 0;
        char c;
        while ((c = getchar()) != '\n' && i < 99) { //getcharë¥¼
í\231\234ì\232©í\225\230ì\227¬ cì\227\220 ë¬¸ì\236\220ë¥¼ ì\235¼ì\235¼ì\235´
ì \200ì\236¥í\225\230ê³ , str[] ë°°ì\227´ì\227\220 ë\204£ì\226´ì¤\200ë\213¤
        str[i++] = c;
        }
        str[i] = '\0';


        if(check_palindrome_array(str))
                printf("Answer(array) : \"%s\" is a palindrome.\n", str);
        else
                printf("Answer(array) : \"%s\" is not a palindrome.\n", str);

        if(check_palindrome_pointer(str))
                printf("Answer(pointer) : \"%s\" is a palindrome.\n", str);
        else
                printf("Answer(pointer) : \"%s\" is not a palindrome.\n", str);




        return 0;
}



int get_length(char *str) { //ë¬¸ì\236\220ì\227´ ê¸¸ì\235´ë¥¼
ì¸ì \225í\225\230ë\212\224 í\225¨ì\210\230 ë\204\220 ë¬¸ì\236\220ê°\200
ì\230¬ë\225\214ê¹\214ì§\200 ì\204¼ë\213¤
        int len = 0;

        while (str[len] != '\0') {
                len++;
        }

        return len;
}


// ì\225\214í\214\214ë²³ì\235¸ì§\200 í\231\225i\235¸í\225\230ë\212\224
í\225¨ì\210\230 aë¶\200í\204° zê¹\214ì§\200ì\231\200, Aë¶\200í\204° Zê¹\214ì§\200
int is_alpha(char c) {
        return (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');
}


// ë\214\200ë¬¸ì\236\220ë¥¼ ì\206\214ë¬¸ì\236\220ë¡\234
ë³\200í\231\230í\225\230ë\212\224 í\225¨ì\210\230
char to_lower(char c) {
        if (c >= 'A' && c <= 'Z') {
                return c + ('a' - 'A'); // 97 - 65 = 32
        }

        return c;
}



// ë°°ì\227´ì\235\204 ì\202¬ì\232©í\225\234 í\214°ë¦°ë\223\234ë¡¬ ê²\200ì\202¬
int check_palindrome_array(char str[]) {
        int len = get_length(str);
        char cleaned[100];
        int j = 0;

        // ì\225\214í\214\214ë²³ë§\214 ì¶\224ì¶\234í\225\230ì\227¬
ì\206\214ë¬¸ì\236\220ë¡\234 ì \200ì\236¥. ì\234\204ì\227\220ì\204\234 ë¬¸ì\236\220ì\227´
ê¸¸ì\235´ë¥¼ ì¸¡ì \225í\225\230ë\212\224 í\225¨ì\210\230ê°\200 ì\236\210ì\234¼ë\202\230,
ì\227¬ê°°ì\204\234ë\212\224 ë¬¸ì\236\220ì\227´ ì¤\221ì\227\220ì\204\234
ì\225\214í\214\214ë²³ë§\214 ì\204 í\203\235ì \201ì\234¼ë¡\234 ê°\234ì\210\230ë¥¼
ì\204¼ë\213¤
        for(int i = 0; i < len; i++) {
                if(is_alpha(str[i])) {
                cleaned[j] = to_lower(str[i]);
                j++;
                }
        }
        cleaned[j] = '\0'; //ë\213¤ ë\201\235ë\202\230ë©´ ë\201\235ì\227\220
nullë¬¸ì\236\220ë¥¼ ë\204£ì\226´ì\204\234 ë§\210ë¬´ë¦¬ë¥¼ ì§\223ë\212\224ë\213¤


        // í\214°ë¦°ë\223\234ë¡¬ ê²\200ì\202¬
        for(int i = 0; i < j/2; i++) {
                if(cleaned[i] != cleaned[j-1-i]) {
                return 0; //ì¤\221ê°\204ì\227\220 ë¶\210ì\235¼ì¹\230ë¥¼
ë°\234ê²¬í\225  ê²½ì\232° í\225¨ì\210\230ë¥¼ ì¦\211ì\213\234 ì¢\205ë£\214
                }
        }
        return 1;//ì\234\204ì\235\230 ì¡°ê±´ì\235\204 ë§\214ì¡±í\225  ê²½ì\232°
í\214°ë¦°ë\223\234ë¡¬ì\235´ë\235¼ê³  í\214\220ì \225
}


// í\217¬ì\235¸í\204°ë¥¼ ì\202¬ì\232©í\225\234 í\214°ë¦°ë\223\234ë¡¬ ê²\200ì\202¬
        int check_palindrome_pointer(char *str) {
        int len = get_length(str);
        char cleaned[100];
        char *ptr = cleaned;

        // ì\225\214í\214\214ë²³ë§\214 ì¶\224ì¶\234í\225\230ì\227¬
ì\206\214ë¬¸ì\236\220ë¡\234 ì \200ì\236¥
        while (*str != '\0') {
                if(is_alpha(*str)) {
                *ptr = to_lower(*str);
                ptr++;
                }
                str++;
        }
        *ptr = '\0';
```

```
  119:         // í\214°ë¦°ë\223\234ë¡¬ ê²\200ì\202¬
  120:         char *start = cleaned;
  121:         char *end = ptr - 1;
  122:
  123:         while(start < end) {
  124:                 if(*start != *end) {
  125:                 return 0; //ë§\210ì°¬ê°\200ì§\200ë¡\234 ë\213¤ë¥¼ ê²½ì\232°
ì¦\211ì\213\234 ì¢\205ë£\214
  126:                 }
  127:                 start++;
  128:                 end--;
  129:         }
  130:         return 1; //ì\234\204ì\235\230 ì¡°ê±´ì\227\220ì\204\234
ì\202´ì\225\204ë\202¨ì\234¼ë©´ í\214°ë¦°ë\223\234ë¡¬ í\214\220ì \225
  131: }
  132:
  133:
```

```c
#include <stdio.h>

// í\225¨ì\210\230 ì\233\220í\230\225 ì\204 ì\226¸ - í\217¬ì\235¸í\204°í\221\234ê¸°ë²\225 ì\202¬ì\232©
void add_matrix(int (*A)[10], int (*B)[10], int (*result)[10], int n);
int get_diagonal_sum(int (*matrix)[10], int n);
int is_sparse_matrix(int (*matrix)[10], int n);

int main(void) {
        int N;
        int check;

        printf("[Input]\n");
        //N ì\236\205ë ¥ê°\222 ê²\200ì¦\235
        while(1) {
                check = scanf("%d", &N);
                if(check != 1) {
                        printf("Invalid input. Please enter a number: ");
                        while(getchar() != '\n');
                        continue;
                }

                if(N >= 2 && N <= 10) {
                break;
                }

                printf("N must be between 2 and 10. Please enter again: ");
                }

        int A[10][10] = {0}, B[10][10] = {0}, result[10][10] = {0};

        // A í\226\211ë ¬ ì\236\205ë ¥
        for(int i = 0; i < N; i++) {
                for(int j = 0; j < N; j++) {
                while(1) {
                        check = scanf("%d", &A[i][j]);
                        if(check != 1) {
                                printf("Invalid input. Please enter a number for A[%d][%d]: ", i, j);
                                while(getchar() != '\n');
                                continue;
                        }

                        if(A[i][j] >= -100 && A[i][j] <= 100) {
                                break;
                        }

                        printf("Element must be between -100 and 100. Please enter again for A[%d][%d]: ", i, j);
                        }
                }
        }

        // B í\226\211ë ¬ ì\236\205ë ¥
        for(int i = 0; i < N; i++) {
                for(int j = 0; j < N; j++) {
                while(1) {
                        check = scanf("%d", &B[i][j]);
                                if(check != 1) {
                                        printf("Invalid input. Please enter a number for B[%d][%d]: ", i, j);
                                        while(getchar() != '\n');
                                        continue;
                                        }

                                if(B[i][j] >= -100 && B[i][j] <= 100) {
                                        break;
                                        }

                                printf("Element must be between -100 and 100. Please enter again for B[%d][%d]: ", i, j);
                                }
                        }
                }
        }

        // í\226\211ë ¬ ë\215§ì\205\210, ë\214\200ê°\201í\225© ê³\204ì\202°, í\235¬ì\206\214í\226\211ë ¬ í\214\220ë³\204
        printf("[Output]\n");
        add_matrix(A, B, result, N);


        // ê²°ê³¼ í\226\211ë ¬ ì¶\234ë ¥
        for(int i = 0; i < N; i++) {
                for(int j = 0; j < N; j++) {
                        printf("%d ", result[i][j]);
                }
                printf("\n");
        }

        printf("%d\n", get_diagonal_sum(result, N));

        printf("A: %s, B: %s\n",
                is_sparse_matrix(A, N) ? "sparse matrix" : "not a sparse matrix",
                is_sparse_matrix(B, N) ? "sparse matrix" : "not a sparse matrix");

        return 0;
}

// í\226\211ë ¬ ë\215§ì\205\210 í\225¨ì\210\230
void add_matrix(int (*A)[10], int (*B)[10], int (*result)[10], int n) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            result[i][j] = A[i][j] + B[i][j];
        }
    }
}

// ë\214\200ê°\201ì\233\220ì\206\214 í\225© ê³\204ì\202° í\225¨ì\210\230
int get_diagonal_sum(int (*matrix)[10], int n) {
    int sum = 0;
    for(int i = 0; i < n; i++) {
        sum += matrix[i][i];
    }
    return sum;
}

// í\235¬ì\206\214í\226\211ë ¬ í\214\220ë³\204 í\225¨ì\210\230
int is_sparse_matrix(int (*matrix)[10], int n) {
    int zero_count = 0;
    int total_elements = n * n;

    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            if(matrix[i][j] == 0) {
                zero_count++;
            }
        }
    }

    return zero_count >= total_elements/2;
}
```