1.

```
. #include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int isValidSegment(const char *segment)
{
    if (*segment == '\0') {
        return 0;
    }
    if (segment[0] == '0' && strlen(segment) > 1)
    {
        return 0;
    }
    int num = atoi(segment);
    if (num < 0 || num > 255) {
        return 0;
    }
    for (int i = 0; segment[i] != '\0'; i++) {
        if (!isdigit(segment[i])) {
            return 0;
        }}
return 1;
}
int isValidIPv4(const char *ip) {
    char ipCopy[16];
    strcpy(ipCopy, ip);
    char *segment = strtok(ipCopy, ".");
    int segmentCount = 0;
    while (segment != NULL) {
    if (!isValidSegment(segment)) {
            return 0;
    }
        segmentCount++;
        segment = strtok(NULL, ".");
    }
    return segmentCount == 4;
}
int main() {
    char ip[16];
    printf("Enter an IPv4 address: ");
    scanf("%15s", ip);
    if (isValidIPv4(ip)) {
        printf("Output: true\n");
    } else {
        printf("Output: false\n");
    }
    return 0;}
```

2.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX_LENGTH 1000
int isAlphanumeric(char c) {
    return (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');
}

// Function to expand around the center and find palindromic substrings
void findPalindromes(char *s, int left, int right, char **palindromes, int *count) {
    while (left >= 0 && right < strlen(s) && s[left] == s[right]) {
        int len = right - left + 1;
        char *substr = (char *)malloc(len + 1);
        strncpy(substr, s + left, len);
        substr[len] = '\0';
        palindromes[*count] = substr;
        (*count)++;

        left--;
        right++;
    }
}
int countDistinctPalindromicSubstrings(char *s) {
    char *palindromes[MAX_LENGTH];
    int count = 0;
    for (int i = 0; i < strlen(s); i++) {
        findPalindromes(s, i, i, palindromes, &count);
        findPalindromes(s, i, i + 1, palindromes, &count);
    }
    int distinctCount = 0;
    for (int i = 0; i < count; i++) {
        int isDistinct = 1;
        for (int j = 0; j < i; j++) {
            if (strcmp(palindromes[i], palindromes[j]) == 0) {
                isDistinct = 0;
                break;
            }
        }
        if (isDistinct) {
            distinctCount++;
        }
        free(palindromes[i]);
    }
    return distinctCount;
}
```

```
int main() {
    char input[MAX_LENGTH];
    printf("Enter a string: ");
    fgets(input, MAX_LENGTH, stdin);
    input[strcspn(input, "\n")] = 0; // Remove newline character
    int result = countDistinctPalindromicSubstrings(input);
   printf("Input: %s\n", input);
    printf("Output: %d\n", result);

    return 0;
}
```

3.

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

void findNextGreater(int *arr, int *result, int size, int index) {
    if (index < 0) {
        return;
    }
    int nextGreater = -1;
    for (int j = index + 1; j < size; j++) {
        if (arr[j] > arr[index]) {
            nextGreater = arr[j];
            break;
        }
    }
    result[index] = nextGreater;
    findNextGreater(arr, result, size, index - 1);
}

void nextGreaterPrices(int *arr, int *result, int size) {
    findNextGreater(arr, result, size, size - 1);
}
int main() {
    int arr[] = {6, 8, 0, 1, 3};
    int size = sizeof(arr) / sizeof(arr[0]);
    int result[size];
    for (int i = 0; i < size; i++) {
        result[i] = -1;
    }
    nextGreaterPrices(arr, result, size);
    printf("Output: [");
    for (int i = 0; i < size; i++) {
        printf("%d", result[i]);
        if (i < size - 1) {
            printf(", ");
        }
    }
    printf("]\n");
    return 0;
}
```