APPENDIX

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

struct Node { char
    word[50];
    char meaning[100]; struct
    Node* next;
};

// Function to create a new node
struct Node* createNode(char word[], char meaning[]) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    strcpy(newNode->word, word);
    strcpy(newNode->meaning, meaning);
    newNode->next = NULL;
    return newNode;
}

// Insert node in alphabetical order
void insert(struct Node** head, char word[], char meaning[]) { struct
    Node* newNode = createNode(word, meaning);

    if (*head == NULL || strcmp((*head)->word, word) > 0) {
        newNode->next = *head;
        *head = newNode;
        return;
    }

    struct Node* current = *head;
    while (current->next != NULL && strcmp(current->next->word, word) < 0) { current =
        current->next;
    }

    newNode->next = current->next;
    current->next = newNode;
}

// Search for a word
void search(struct Node* head, char word[]) { while (head
    != NULL) {
        if (strcmp(head->word, word) == 0) {
            printf("Meaning: %s\n", head->meaning); return;
        }
        head = head->next;
    }
    printf("Word not found in dictionary.\n");
}

// Display all words
void display(struct Node* head) { if
    (head == NULL) {
        printf("Dictionary is empty.\n"); return;
    }

    printf("\nDictionary:\n"); while
    (head != NULL) {
        printf("Word: %-15s Meaning: %s\n", head->word, head->meaning); head = head-
        >next;
    }
}

// Main function int
main() {
    struct Node* dictionary = NULL; int
    choice;
    char word[50], meaning[100];

    do {
        printf("\n--- Word Dictionary Menu ---\n");
        printf("1. Add Word\n2. Search Word\n3. Display All\n4. Exit\n"); printf("Enter choice: ");
        scanf("%d", &choice); getchar();  //
        Clear newline

        switch (choice) { case 1:
                printf("Enter word: "); fgets(word,
                sizeof(word), stdin);
                word[strcspn(word, "\n")] = '\0';  // Remove newline
                printf("Enter meaning: ");
                fgets(meaning, sizeof(meaning), stdin);
                meaning[strcspn(meaning,
                "\n")] = '\0';
                insert(&dictionary, word,
                meaning); break;

            case 2:
                printf("Enter word to
                search:                 ");
                fgets(word,
                sizeof(word),  stdin);
                word[strcspn(word,
                "\n")]       =       '\0';
                search(dictionary,
                word);
                break;

            case 3:
                display(dicti
                onary);
                break;

            case 4:
                printf("Exiting
                dictionary.\n");
                break;

            default:
                printf("Invalid choice.\n");
        }
    } while (choice != 4);

    return 0;
}
```