भारतीय सूचना प्रौद्योगिकी संस्थान गुवाहाटी

Indian Institute of Information Technology Guwahati

COMPUTER PROGRAMMING LAB (CS110)

SOLUTIONS–12

1. Consider a singly-linked list that stores integers. Implement the following functions in C:

   i. Write a function to add a node at the beginning of the list to store a given integer.

   ii. Write a function to add a node at the end of the list to store a given integer.

   iii. Write a function to add a node at the ith node of the list to store a given integer.

   iv. Write a function to delete the first node of the list returning the stored integer in the node.

   v. Write a function to delete the last node of the list returning the stored integer in the node.

   vi. Write a function to delete the ith node of the list returning the stored integer in the node.

   vii. Write a function to print the list.

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

typedef struct Node_t {
    int data;
    struct Node_t *next;
} Node_t, *Node;

Node newNode(int data, Node next) {
    Node node = (Node) calloc(1, sizeof (Node_t));
    node->data = data;
    node->next = next;
    return node;
}

void addFirst(Node *pnode, int data) {
    *pnode = newNode(data, *pnode);
```

```
}

void addLast(Node *pnode, int data) {
    if (*pnode == NULL) {
        addFirst(pnode, data);
        return;
    }
    Node node = *pnode;
    while (node->next)
        node = node->next;
    node->next = newNode(data, node->next);
}

int addAt(Node *pnode, int data, int index) {
    if (index == 0) {
        addFirst(pnode, data);
        return 1;
    }
    Node node = *pnode;
    for (index--; node && index; index--)
        node = node->next;
    if (!node) return 0;
    node->next = newNode(data, node->next);
}

int removeFirst(Node *pnode) {
    if (*pnode == NULL)
        return INT_MAX;
    Node node = *pnode;
    *pnode = node->next;
    int data = node->data;
    free(node);
    return data;
}

int removeLast(Node *pnode) {
    if (*pnode == NULL || (*pnode)->next == NULL)
        removeFirst(pnode);
    Node node = *pnode;
    while (node->next->next)
        node = node->next;
    int data = node->next->data;
    free(node->next);
    node->next = NULL;
    return data;
}

int removeFrom(Node *pnode, int index) {
    if (index == 0)
        return removeFirst(pnode);
```

```
    Node node = *pnode;
    for (index--; index; index--)
        node = node->next;
    if (node->next == NULL)
        return INT_MAX;
    int data = node->next->data;
    Node temp = node->next;
    node->next = node->next->next;
    free(temp);
    return data;
}

void printList(Node node) {
    while (node) {
        printf("%d -> ", node->data);
        node = node->next;
    }
    printf("NULL\n");
}

int main() {
    Node list = NULL;
    printList(list);
    addFirst(&list, 10);
    printList(list);
    addFirst(&list, 9);
    printList(list);
    addFirst(&list, 8);
    printList(list);
    addLast(&list, 11);
    printList(list);
    addLast(&list, 12);
    printList(list);
    addLast(&list, 13);
    printList(list);
    int data = removeFirst(&list);
    printf("Removed: %d\n", data);
    printList(list);
    data = removeLast(&list);
    printf("Removed: %d\n", data);
    printList(list);
    int index = 0;
    data = 8;
    if (addAt(&list, data, index))
        printf("Added %d at %d.\n", data, index);
    else
        printf("Failed to add %d at %d.\n", data, index);
    printList(list);
    index = 15;
    data = 13;
```

```
    if (addAt(&list, data, index))
        printf("Added %d at %d.\n", data, index);
    else
        printf("Failed to add %d at %d.\n", data, index);
    printList(list);
    index = 5;
    data = 13;
    if (addAt(&list, data, index))
        printf("Added %d at %d.\n", data, index);
    else
        printf("Failed to add %d at %d.\n", data, index);
    printList(list);
    index = 5;
    data = removeFrom(&list, index);
    if (data != INT_MAX)
        printf("Added %d from %d.\n", data, index);
    else
        printf("Failed to add %d from %d.\n", data, index);
    printList(list);
    index = 0;
    data = removeFrom(&list, index);
    if (data != INT_MAX)
        printf("Added %d from %d.\n", data, index);
    else
        printf("Failed to add %d from %d.\n", data, index);
    printList(list);
    index = 2;
    data = removeFrom(&list, index);
    if (data != INT_MAX)
        printf("Added %d from %d.\n", data, index);
    else
        printf("Failed to add %d from %d.\n", data, index);
    printList(list);
}
```

2. Write a program in C to copy a source text file to a target text file. The source and the target filenames should be command-line arguments.
Solution:

```
/**
Do not use goto. However, if you choose to use goto, this program
shows a possibly good example of using goto for exception handling.
*/
#include <stdio.h>

int main(int argc, char *argv[]) {
    if (argc != 3) goto ImproperArgumentsException;

    char *source = argv[1], *target = argv[2];
```

```
    FILE *reader = fopen(source, "r");
    if (!reader) goto FileReadException;

    FILE *writer = fopen(target, "w");
    if (!writer) goto FileWriteException;

    for (char c = 0; (c = fgetc(reader)) != EOF; fputc(c, writer))
        ;

    fclose(reader);
    fflush(writer);
    fclose(writer);

    fprintf(stdout, "%s is successfully copied to %s.\n", source, target);

    return 0;

    ImproperArgumentsException:
        fprintf(stdout, "Usage: <executable> <source_file> <target_file>\n");
        return -1;

    FileReadException:
        fprintf(stderr, "Exception in reading file: %s\n", source);
        fprintf(stderr, "The program will now terminate.\n");
        return -1;

    FileWriteException:
        fclose(reader);
        fprintf(stderr, "Exception in writing file: %s\n", target);
        fprintf(stderr, "The program will now terminate.\n");
        return -1;
}
```

3. Write a program in C to copy a source file to a target file (open the both the files in binary mode). The source and the target filenames should be command-line arguments.
Solution:

```
/**
Do not use goto. However, if you choose to use goto, this program
shows a possibly good example of using goto for exception handling.
*/
#include <stdio.h>

int main(int argc, char *argv[]) {
    if (argc != 3) goto ImproperArgumentsException;

    char *source = argv[1], *target = argv[2];
```

```
    FILE *reader = fopen(source, "rb");
    if (!reader) goto FileReadException;

    FILE *writer = fopen(target, "wb");
    if (!writer) goto FileWriteException;

    char c = 0;
    while (fread(&c, sizeof (char), 1, reader))
        fwrite(&c, sizeof (char), 1, writer);

    fclose(reader);
    fflush(writer);
    fclose(writer);

    fprintf(stdout, "%s is successfully copied to %s.\n", source, target);

    return 0;

    ImproperArgumentsException:
        fprintf(stdout, "Usage: <executable> <source_file> <target_file>\n");
        return -1;

    FileReadException:
        fprintf(stderr, "Exception in reading file: %s\n", source);
        fprintf(stderr, "The program will now terminate.\n");
        return -1;

    FileWriteException:
        fclose(reader);
        fprintf(stderr, "Exception in writing file: %s\n", target);
        fprintf(stderr, "The program will now terminate.\n");
        return -1;
}
```

4. Realize the following program:

```
# include <stdio.h>

typedef float (*FloatFunctionFloatFloat) (float, float);

float add(float x, float y) {
    return x + y;
}

float sub(float x, float y) {
    return x - y;
}

float mul(float x, float y) {
    return x * y;
```

```
}

float div(float x, float y) {
    return x / y;
}

FloatFunctionFloatFloat inverse(FloatFunctionFloatFloat function) {
    if (function == add) {
        return sub;
    }
    if (function == sub) {
        return add;
    }
    if (function == div) {
        return mul;
    }
    if (function == mul) {
        return div;
    }
    return NULL;
}

int main() {
    float x = 6, y = 4;
    float z = (mul - add + sub)(x, y); // calls div(x, y)
    printf("(mul - add + sub)(%g, %g) = %g\n", x, y, z);
    printf("(mul - add + sub) = %p, div = %p\n", mul - add + sub, div);

    FloatFunctionFloatFloat f = add;
    float w = inverse(f)(f(x, y), y); // calls sub(add(x, y), y)
    printf("inverse(f)(f(%g, %g), %g) = %g\n", x, y, y, w);

    return 0;
}
```

5. Realize the following program:

```
#include <stdio.h>

#define SIZE 10

int main() {
    //anonymous structure
    struct {int x; int y;} a = {
        .y = 7,
        .x = 6,
    };
    printf("a.x = %d, a.y = %d\n", a.x, a.y);

    // Less popular way of sparse array initialization
```

```
    int array[SIZE] = {[5] 7, 19, [3] 17, 18};
    for (int i = 0; i < SIZE; i++) {
        printf("%d ", array[i]);
    }
}
```

6. Realize the following program:

```
# include <stdio.h>

typedef double  (*DoubleFunctionDouble)(double);

double add(double x, double y) {
    return x + y;
}

DoubleFunctionDouble curryAdd(double x) { // currying
    double f(double y) { // nested function; not allowed in C; GCC extension
        return x + y;
    }
    return f;
}

int main(int argc, char *argv[]) {
    printf("%g\n", add(3.2, 5.6));
    printf("%g\n", curryAdd(3.2)(5.6));
    return 0;
}
```

7. Realize the following program:

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

double add(const char *format, ...) { // function taking variable number of arguments
    double total = 0.0;
    va_list list;
    va_start(list, format);
    for (int i = 0; format[i]!= '\0'; i++) {
        int s = format[i];
        switch (s) {
            case 'c' : // char is promoted to int va_list
            case 'i' : total += va_arg(list, int);
            break;
            case 'f' : // float is promoted to double in va_list
            case 'd' : total += va_arg(list, double);
            break;
            default  : break;
        }
    }
```

```
    return total;
}

int main(int argc, char *argv[]) { // Pseudo function overloading in C
    printf("add(\"c\", 'a') = %lg\n", add("c", 'a'));
    printf("add(\"cf\", 'a', 1.0) = %lg\n",  add("cf", 'a', 1.0));
    printf("add(\"cid\", 'a', 1, 2.0) = %lg\n", add("cid", 'a', 1, 2.0));
    return 0;
}
```