

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define item in linked-list:

typedef struct node {

int num;

struct node \*next;

};

int menu\_select();

node \*Add\_end(node \*head, int input);

void print\_list(node \*head);

int count\_list(node \*head);

void free\_list(node \*head);

int sum\_list\_nodes(node \*head);

node \*combine\_lists(node \*list1, node \*list2);

int main()

{

int num = 0, count, sumList1 = 0, sumList2 = 0, choice, input = 0;

node \*head1 = NULL, \*head2 = NULL;

for (;;) // infinite for loop

{

choice = menu\_select();

switch (choice)

{

case 1:

printf("Enter number to create new node:\n");

scanf\_s("%d", &input);

while (input != -1) {

head1 = Add\_end(head1, input);

printf("Enter number to create new node:(-1 to exit)\n");

scanf\_s("%d", &input);

} break;

case 2:

printf("List 1:\n");

print\_list(head1);

count = count\_list(head1);

printf(" List 1 has %d nodes\n", count);

break;

case 3: free\_list(head1);

break;

case 4:

printf("Enter number to create new node:\n");

scanf\_s("%d", &input);

while (input != -1) {

head2 = Add\_end(head2, input);

printf("Enter number to create new node:(-1 to exit)\n");

scanf\_s("%d", &input);

} break;

case 5:

printf("List 2:\n");

print\_list(head2);

count = count\_list(head2);

printf(" List 2 has %d nodes\n", count);

break;

case 6: free\_list(head1); break;

case 7:

sumList1 = sum\_list\_nodes(head1);

printf(" List 1 sum nodes: %d\n", sumList1);

sumList2 = sum\_list\_nodes(head2);

printf(" List 2 sum nodes: %d\n", sumList2);

case 8:

combine\_lists(head1, head2);

printf("List 1 and list 2 comblined:\n");

break;

case 9:

printf("New combined list:");

if (sumList1 > sumList2)

print\_list(head1);

else

print\_list(head2);

break;

case 10:

free\_list(head1); break;

case 11:

break;

}

}

system("cls");

getchar();

return 0;

}

// menu:

int menu\_select() {

int c;

printf("--------------LIST - 1 --------------------\n");

printf(" 1. Add new node to List 1\n");

printf(" 2. Print list 1 + conut nodes\n");

printf(" 3. free list\n");

printf("--------------LIST - 2 --------------------\n");

printf(" 4. Add new node to List 2\n");

printf(" 5. Print list 2 + conut nodes\n");

printf(" 6. free list\n");

printf("-------------Check list sum: --------------\n");

printf(" 7. Compare list nodes sum\n");

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

printf(" 8. Combine List1& List2\n");

printf(" 9. Print new combined list\n");

printf(" 10. Free new list\n");

printf(" 11. Exit program\n");

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

printf("Enter your choice:\n");

scanf\_s("%d", &c);

if(c>0 && c<=11)

return c;

}

node \*Add\_end(node \*head, int input) {

node \*tail = head;

node \*item = (node\*)malloc(sizeof(node));

item->num = input;

item->next = NULL;

if (tail == NULL)

return item;

while (tail->next != NULL)

tail = tail->next;

tail->next = item;

return head;

}

void print\_list(node \*head) {

node \*ptr = NULL;

for (ptr = head; ptr != NULL; ptr = ptr->next)

printf("%d\n", ptr->num);

}

int count\_list(node \*head) {

int sum = 0;

while (head != NULL) {

sum ++;

head = head->next;

}

return sum;

}

void free\_list(node \*head) {

node \*ptr = head;

while (ptr != NULL) {

head = head->next;

free(ptr);

ptr = head;

}

printf("List is free and deleted!!!\n");

}

int sum\_list\_nodes(node \*head) {

int sum=0;

while (head != NULL)

{

sum += head->num;

head = head->next;

}

return sum;

}

node \*combine\_lists(node \*list1, node \*list2)

{

node \*list1\_head = list1, \*list2\_head = list2;

int sumList1 = 0, sumList2 = 0;

// Sum of list 1:

while (list1\_head != NULL) {

sumList1 += list1\_head->num;

list1\_head = list1\_head->next;

}

// Sum of list 2:

while (list2\_head != NULL) {

sumList2 += list2\_head->num;

list2\_head = list2\_head->next;

}

// Check list sum for concatination:

if (sumList1 > sumList2)

{

while (list1->next != NULL)

list1 = list1->next;

list1->next = list2;

while (list2->next != NULL)

list2 = list2->next;

list2->next = list1\_head;

return list1\_head;

}

else {

while (list2->next != NULL)

list2 = list2->next;

list2->next = list1;

while (list1->next != NULL)

list1 = list1->next;

list1->next = list2\_head;

return list2\_head;

}

}