WEEK-5: Test : @ 16.3.2024 / 18.3.2024

Q: Write C functions to implement insert at begin, end and given position in SLL

TURBO C++ Practice Programs

**//DLL Implementation**

**#include<stdio.h>**

**#include<conio.h>**

**#include<alloc.h>**

**struct dll**

**{**

**struct dll \*llink;**

**int data;**

**struct dll \*rlink;**

**};**

**typedef struct dll node;**

**node \*first=NULL,\*temp,\*newnode;**

**void create();**

**void display();**

**void search();**

**void count();**

**void insert();**

**void insertbgn();**

**void insertpos();**

**void insertend();**

**void del();**

**void delbgn();**

**void delpos();**

**void delend();**

**void main()**

**{**

**char ch='y';**

**int option;**

**clrscr();**

**while(ch=='y'|| ch=='Y')**

**{**

**printf("\n1.create\n2.display\n3.insert\n4.delete\n5.search\n6.count\n");**

**printf("enter ur option\n");**

**scanf("%d",&option);**

**switch(option)**

**{**

**case 1: create();**

**break;**

**case 2: display();**

**break;**

**case 3: insert();**

**break;**

**case 4: del();**

**break;**

**case 5: search();**

**break;**

**case 6: count();**

**break;**

**}**

**printf("do u want to continue with any other option(Y/N)\n");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void create()**

**{**

**char ch='y';**

**while(ch=='y' || ch=='Y')**

**{**

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

**printf("enter data for new node\n");**

**scanf("%d",&newnode->data);**

**if(first==NULL)**

**{**

**first=newnode;**

**newnode->llink=NULL;**

**newnode->rlink=NULL;**

**}**

**else**

**{**

**temp=first;**

**while(temp->rlink!=NULL)**

**temp=temp->rlink;**

**temp->rlink=newnode;**

**newnode->llink=temp;**

**newnode->rlink=NULL;**

**}**

**printf("do u want to create another node(Y/N)\n");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void display()**

**{**

**if(first==NULL)**

**{**

**printf("list is empty");**

**exit();**

**}**

**else**

**{**

**printf("elements in the list\n");**

**temp=first;**

**while(temp->rlink!=NULL)**

**{**

**printf("%d->", temp->data);**

**temp=temp->rlink;**

**}**

**printf("%d\n",temp->data);**

**}**

**}**

**void insert()**

**{**

**char ch='y';**

**int c;**

**while(ch=='y'||ch=='Y')**

**{**

**printf("\n 1.insert at begining\n 2.insert at pos.n\n 3.insert at end\n");**

**printf("enter ur option\n");**

**scanf("%d",&c);**

**switch(c)**

**{**

**case 1: insertbgn();**

**break;**

**case 2: insertpos();**

**break;**

**case 3: insertend();**

**break;**

**}**

**printf("do u want to insert one more node(y/n)\n");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void insertbgn()**

**{**

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

**printf("Enter data for newnode\n");**

**scanf("%d",&newnode->data);**

**if(first==NULL)**

**{**

**first=newnode;**

**newnode->llink=NULL;**

**newnode->rlink=NULL;**

**}**

**else**

**{**

**newnode->rlink=first;**

**first->llink=newnode;**

**newnode->llink=NULL;**

**first=newnode;**

**}**

**}**

**void insertpos()**

**{**

**int i=2,pos;**

**printf("enter the position to insert the node\n");**

**scanf("%d",&pos);**

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

**printf("enter data for newnode\n");**

**scanf("%d",&newnode->data);**

**temp=first;**

**while(i<pos)**

**{**

**temp=temp->rlink;**

**i++;**

**}**

**temp->rlink->llink=newnode;**

**newnode->rlink=temp->rlink;**

**temp->rlink=newnode;**

**newnode->llink=temp;**

**}**

**void insertend()**

**{**

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

**printf("enter data for newnode\n");**

**scanf("%d",&newnode->data);**

**temp=first;**

**while(temp->rlink!=NULL)**

**{**

**temp=temp->rlink;**

**}**

**temp->rlink=newnode;**

**newnode->llink=temp;**

**newnode->rlink=NULL;**

**}**

**void del()**

**{**

**char ch='y';**

**int c;**

**while(ch=='y'||ch=='Y')**

**{**

**printf("1.delete at begining\n2.delete at req pos\n3.delete at end\n");**

**printf("enter your option\n");**

**scanf("%d",&c);**

**switch(c)**

**{**

**case 1: delbgn();**

**break;**

**case 2: delpos();**

**break;**

**case 3: delend();**

**break;**

**}**

**printf("do u want to deelete another node(y/n)");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void delbgn()**

**{**

**if(first==NULL)**

**{**

**printf("deletion not possible\n");**

**exit();**

**}**

**else**

**{**

**temp=first;**

**first=first->rlink;**

**first->llink=NULL;**

**free(temp);**

**}**

**}**

**void delpos()**

**{**

**int pos,i=2;**

**temp=first;**

**printf("Enter the position of the node to be deleted\n");**

**scanf("%d",&pos);**

**while(i<pos)**

**{**

**temp=temp->rlink;**

**i++;**

**}**

**temp->rlink->rlink->llink=temp;**

**temp->rlink=temp->rlink->rlink;**

**}**

**void delend()**

**{**

**temp=first;**

**while(temp->rlink->rlink!=NULL)**

**{**

**temp=temp->rlink;**

**}**

**temp->rlink=NULL;**

**}**

**void search()**

**{**

**int item,flag=0,i=1;**

**temp=first;**

**printf("enter the element that u want to search \n");**

**scanf("%d",&item);**

**while(temp!=NULL)**

**{**

**if(item==temp->data)**

**{**

**flag=1;**

**break;**

**}**

**temp=temp->rlink;**

**i++;**

**}**

**if(flag==1)**

**printf("element %d is present at location%d\n",item,i++);**

**else**

**printf("element %d not found in the list\n",item);**

**}**

**void count()**

**{**

**int c=0;**

**temp=first;**

**while(temp!=NULL)**

**{**

**temp=temp->rlink;**

**c++;**

**}**

**printf("Number of nodes in the list are %d \n",c);**

**} \*/**

**// CLL Implementation**

**#include<stdio.h>**

**#include<conio.h>**

**#include<alloc.h>**

**struct csll**

**{**

**int data;**

**struct csll \*next;**

**};**

**typedef struct csll node;**

**node \*head=NULL,\*temp,\*newnode;**

**void create();**

**void display();**

**void search();**

**void count();**

**void insert();**

**void insertbgn();**

**void insertpos();**

**void insertend();**

**void del();**

**void delbgn();**

**void delpos();**

**void delend();**

**void main()**

**{**

**char ch='y';**

**int option;**

**clrscr();**

**while(ch=='y'|| ch=='Y')**

**{**

**printf("\n1.create\n2.display\n3.insert\n4.delete\n5.search\n6.count\n");**

**printf("enter ur option\n");**

**scanf("%d",&option);**

**switch(option)**

**{**

**case 1: create();**

**break;**

**case 2: display();**

**break;**

**case 3: insert();**

**break;**

**case 4: del();**

**break;**

**case 5: search();**

**break;**

**case 6: count();**

**break;**

**}**

**printf("do u want to continue with any other option(Y/N)\n");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void create()**

**{**

**char ch='y';**

**while(ch=='y' || ch=='Y')**

**{**

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

**printf("enter data for new node\n");**

**scanf("%d",&newnode->data);**

**if(head==NULL)**

**{**

**head=newnode;**

**newnode->next=head;**

**}**

**else**

**{**

**temp=head;**

**while(temp->next!=head)**

**temp=temp->next;**

**temp->next=newnode;**

**newnode->next=head;**

**}**

**printf("Do u want to create another node(Y/N)\n");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void display()**

**{**

**if(head==NULL)**

**{**

**printf("list is empty");**

**exit(1);**

**}**

**else**

**{**

**printf("elements in the list\n");**

**temp=head;**

**do**

**{**

**printf("%d->", temp->data);**

**temp=temp->next;**

**}while(temp!=head);**

**printf("%d",temp->data);**

**}**

**}**

**void insert()**

**{**

**char ch='y';**

**int c;**

**while(ch=='y'|| ch=='Y')**

**{**

**printf("\n 1.insert at begining\n 2.insert at position\n 3.insert at end\n");**

**printf("enter ur option\n");**

**scanf("%d",&c);**

**switch(c)**

**{**

**case 1: insertbgn();**

**break;**

**case 2: insertpos();**

**break;**

**case 3: insertend();**

**break;**

**}**

**printf("do u want to insert one more node(y/n)\n");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void insertbgn()**

**{**

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

**printf("Enter data for newnode\n");**

**scanf("%d",&newnode->data);**

**if(head==NULL)**

**{**

**head=newnode;**

**newnode->next=head;**

**}**

**else**

**{**

**temp=head;**

**while(temp->next!=head)**

**temp=temp->next;**

**temp->next=newnode;**

**newnode->next=head;**

**head=newnode;**

**}**

**}**

**void insertpos()**

**{**

**int i=2,pos;**

**printf("enter the position to insert the node\n");**

**scanf("%d",&pos);**

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

**printf("enter data for newnode\n");**

**scanf("%d",&newnode->data);**

**temp=head;**

**while(i<pos)**

**{**

**temp=temp->next;**

**i++;**

**}**

**newnode->next=temp->next;**

**temp->next=newnode;**

**}**

**void insertend()**

**{**

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

**printf("enter data for newnode\n");**

**scanf("%d",&newnode->data);**

**temp=head;**

**while(temp->next!=head)**

**{**

**temp=temp->next;**

**}**

**temp->next=newnode;**

**newnode->next=head;**

**}**

**void del()**

**{**

**char ch='y';**

**int c;**

**while(ch=='y'||ch=='Y')**

**{**

**printf("1.delete at begining\n2.delete at req pos\n3.delete at end\n");**

**printf("enter your option\n");**

**scanf("%d",&c);**

**switch(c)**

**{**

**case 1: delbgn();**

**break;**

**case 2: delpos();**

**break;**

**case 3: delend();**

**break;**

**}**

**printf("do u want to delete another node(y/n)");**

**fflush(stdin);**

**ch=getchar();**

**}**

**}**

**void delbgn()**

**{**

**if(head==NULL)**

**{**

**printf("deletion not possible\n");**

**exit(1);**

**}**

**else**

**{**

**temp=head;**

**while(temp->next!=head)**

**temp=temp->next;**

**temp->next=head->next;**

**head=head->next;**

**}**

**}**

**void delpos()**

**{**

**int pos,i=2;**

**temp=first;**

**printf("Enter the position of the node to be deleted\n");**

**scanf("%d",&pos);**

**while(i<pos)**

**{**

**temp=temp->next;**

**i++;**

**}**

**temp->next=temp->next->next;**

**}**

**void delend()**

**{**

**temp=head;**

**while(temp->next->next!=head)**

**{**

**temp=temp->next;**

**}**

**temp->next=head;**

**}**

**void search()**

**{**

**int item,flag=0,i=1;**

**temp=heaad;**

**printf("enter the element that u want to search \n");**

**scanf("%d",&item);**

**do**

**{**

**if(item==temp->data)**

**{**

**flag=1;**

**break;**

**}**

**temp=temp->next;**

**i++;**

**}while(temp!=head);**

**if(flag==1)**

**printf("element %d is present at location%d\n",item,i++);**

**else**

**printf("element %d not found in the list\n",item);**

**}**

**void count()**

**{**

**int c=0;**

**temp=head;**

**do**

**{**

**temp=temp->next;**

**c++;**

**}while(temp!=head);**

**printf("Number of nodes in the list are %d \n",c);**

**}**

**EXAMLY Practice Programs:**

**DATA STRUCTURES\_CODING\_WEEK 5**

**Section 1 - CODING**

**Section Summary**

* No. of Questions: 2 Duration: 180 min

Q1.**Problem Statement**

Write a program to perform the following operations in a singly-linked list.

1. Insert the elements at the beginning
2. Insert the elements at the end
3. Insert the elements at the given position (Position starts from 1)

Insert the elements in the desired position and display the list after insertion.

**Input Format**

The input consists of the following choices as integers.

1. Insert at the Beginning
2. Insert at the End
3. Insert at Position
4. Print Linked List
5. Exit

For options 1 and 2, the input consists of an integer representing the data separated by a space from the option.

For option 3, the input consists of two integers: the position and the element to be inserted separated by a space from the option.

Option 4 prints the linked list.

Option 5 is mandatory indicating the termination.

**Output Format**

For option 4, the output displays the elements present in the linked list separated by space.

For option 5, the output displays "Exiting" indicating the termination of the program.

If the choice is invalid, the output displays "Wrong choice".

**Refer to the sample output for formatting specifications.**

**Constraints**

Position starts from 1.

Valid positions must be given as input.

**Sample Input Sample Output**

1 10

1 20

2 30

3 2 25

1 37

2 50

4

5

37 20 25 10 30 50

Exiting

Q1**Test Solution**

#include <stdio.h>

#define MAX\_SIZE 100 // Maximum size of the array

struct myNode {

int val;

int next;

};

int freeIndex = 0; // Next free index in the array

void insertAtEnd(struct myNode \*list, int \*head, int val) {

int tmpIndex = freeIndex++;

list[tmpIndex].val = val;

list[tmpIndex].next = -1;

if (\*head == -1) {

\*head = tmpIndex;

} else {

int curr = \*head;

while (list[curr].next != -1) {

curr = list[curr].next;

}

list[curr].next = tmpIndex;

}

}

void insertAtBeginning(struct myNode \*list, int \*head, int val) {

int tmpIndex = freeIndex++;

list[tmpIndex].val = val;

list[tmpIndex].next = \*head;

\*head = tmpIndex;

}

void insertAtPos(struct myNode \*list, int \*head, int pos, int new\_val) {

int tmpIndex = freeIndex++;

list[tmpIndex].val = new\_val;

list[tmpIndex].next = -1;

int i = 1;

int curr = \*head;

int last = -1;

while (i < pos) {

last = curr;

curr = list[curr].next;

i++;

}

if (last != -1) {

list[last].next = tmpIndex;

} else {

\*head = tmpIndex;

}

list[tmpIndex].next = curr;

}

void print(struct myNode \*list, int head) {

if (head == -1)

return;

int curr = head;

printf("%d ", list[curr].val);

print(list, list[curr].next);

}

int main() {

int choice, new\_val, pos;

struct myNode myList[MAX\_SIZE];

int myListHead = -1;

while (1) {

scanf("%d", &choice);

switch (choice) {

case 1:

scanf("%d", &new\_val);

insertAtBeginning(myList, &myListHead, new\_val);

break;

case 2:

scanf("%d", &new\_val);

insertAtEnd(myList, &myListHead, new\_val);

break;

case 3:

scanf("%d", &pos);

scanf("%d", &new\_val);

insertAtPos(myList, &myListHead, pos, new\_val);

break;

case 4:

print(myList, myListHead);

printf("\n");

break;

case 5:

printf("Exiting");

return 0;

default:

printf("Wrong choice\n");

break;

}

}

}

Q2.**Problem Statement**

Write a program to reorder the given Linked list. Given a singly linked list*: A*0→*A*1→…→*An*-1→*A*n, reorder it to: *A*0→*An*→*A*1→*An*-1→*A*2→*An*-2→…

﻿

**For example:** Given list is 1->2->3->4->5, the reordered list is 1->5->2->4->3.

**Input Format**

The first line contains an integer n, representing the number of elements in the linked list.

The second line contains n space-separated integers, representing the elements of the linked list.

**Output Format**

The first line of output should print the original linked list in the format: [element\_1] ->[element\_2]-> ... ->[element\_n].

The second line of output should print the rearranged linked list in the format: [element\_1] ->[element\_2] -> ... ->[element\_n].

**Refer to the sample output for the formatting specifications.**

**Constraints**

In this scenario, the given test cases will fall under the following constraints:

3 ≤ n ≤ 20

1 <= Each element in the linked list <= 105.

**Sample Input Sample Output**

5

1 2 3 4 5

1 ->2 ->3 ->4 ->5

1 ->5 ->2 ->4 ->3

Q2**Test Solution**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* newNode(int key) {

struct Node\* temp = (struct Node\*)malloc(sizeof(struct Node));

temp->data = key;

temp->next = NULL;

return temp;

}

void reverselist(struct Node\*\* head) {

struct Node\* prev = NULL, \*curr = \*head, \*next;

while (curr) {

next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

}

\*head = prev;

}

void printlist(struct Node\* head) {

while (head != NULL) {

printf("%d ", head->data);

if (head->next)

printf("->");

head = head->next;

}

printf("\n");

}

void rearrange(struct Node\*\* head) {

struct Node\* slow = \*head, \*fast = slow->next;

while (fast && fast->next) {

slow = slow->next;

fast = fast->next->next;

}

struct Node\* head1 = \*head;

struct Node\* head2 = slow->next;

slow->next = NULL;

reverselist(&head2);

\*head = newNode(0);

struct Node\* curr = \*head;

while (head1 || head2) {

if (head1) {

curr->next = head1;

curr = curr->next;

head1 = head1->next;

}

if (head2) {

curr->next = head2;

curr = curr->next;

head2 = head2->next;

}

}

\*head = (\*head)->next;

}

int main() {

int n, data;

scanf("%d%d", &n, &data);

struct Node\* head = newNode(data);

struct Node\* temp = head;

for (int i = 0; i < n - 1; i++) {

scanf("%d", &data);

temp->next = newNode(data);

temp = temp->next;

}

printlist(head);

rearrange(&head);

printlist(head);

return 0;

}

Top of Form

**DATA STRUCTURES\_ASSESSMENT\_WEEK 5**

**Section 1 - Coding**

**Section Summary**

* No. of Questions: 2 Duration: 90 min

Q1.**Problem Statement**

Velu is working on a project that involves managing a singly linked list data structure. The project requires implementing basic operations on the linked list, including appending new nodes and deleting the last node from the list.

Write a program to achieve the above objectives.

**Example:**

**Input**

10

2 25 15 34 55 73 95 82 52 74

**Output**

2 25 15 34 55 73 95 82 52

**Explanation:** The last node of the linked list is 74. The list after deleting the last node is 2 25 15 34 55 73 95 82 52.

**Input Format**

The first line input consists of an integer **n**, representing the number of nodes to be appended to the linked list.

The second line input consists of **n** space-separated integer values representing each node of the list.

**Output Format**

The output prints the values of the nodes in the modified linked list after deleting the last node.

**Refer to the sample output for the formatting specifications.**

**Constraints**

In this scenario, the given test cases will fall under the following constraints:

1 <= n <= 20

1 <= node values <= 2000

**Sample Input Sample Output**

10

2 25 15 34 55 73 95 82 52 74

2 25 15 34 55 73 95 82 52

Q1**Test Solution**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int val;

struct Node\* next;

};

void append(struct Node\*\* head, int val) {

struct Node\* tmp = (struct Node\*)malloc(sizeof(struct Node));

tmp->val = val;

tmp->next = NULL;

if (\*head == NULL) {

\*head = tmp;

} else {

struct Node\* curr = \*head;

while (curr->next != NULL) {

curr = curr->next;

}

curr->next = tmp;

}

}

void print(struct Node\* head) {

if (head == NULL) {

return;

}

struct Node\* curr = head;

while (curr != NULL) {

printf("%d ", curr->val);

curr = curr->next;

}

}

void delete\_last(struct Node\*\* head) {

if (\*head == NULL || (\*head)->next == NULL) {

free(\*head);

\*head = NULL;

} else {

struct Node\* curr = \*head;

while (curr->next->next != NULL) {

curr = curr->next;

}

free(curr->next);

curr->next = NULL;

}

}

int main() {

int num\_of\_nodes, i, val;

scanf("%d", &num\_of\_nodes);

struct Node\* myList = NULL;

for (i = 0; i < num\_of\_nodes; i++) {

scanf("%d", &val);

append(&myList, val);

}

delete\_last(&myList);

print(myList);

return 0;

}

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form