Assignment Of Day 4

Name : Gayatri Vaibhav Surve.

Gmail id : [gayatrisurve91@gmail.com](mailto:gayatrisurve91@gmail.com)

**Question 1**

**Write a function “insert\_any()” for inserting a node at any given position of the linked list. Assume**

**position starts at 0.**

**Question 2**

**Write a function “delete\_beg()” for deleting a node from the beginning of the linked list.**

**Question 3**

**Write a function “delete\_end()” for deleting a node from the end of the linked list.**

**Question 4**

**In the Binary Search algorithm, it is suggested to calculate the mid as beg + (end - beg) / 2**

**instead of (beg + end) / 2. Why is it so?**

**Question 5**

**Write the algorithm/function for Ternary Search.**

**Answers**

**Question (1),(2),(3)**

#include<stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

}\*head=NULL;

struct node\* create\_list(struct node\*head)

{

int num;

printf("\n Please enter the number : ");

scanf("%d",&num);

struct node \*new,\*q;

new=(struct node \*)malloc(sizeof(struct node));

new->data=num;

new->next=NULL;

if(head==NULL)

{

head=new;

}

else{

q=head;

while(q->next!=NULL)

{

q=q->next;

}

q->next=new;

}

return head;

}

void display(struct node\*head)

{

struct node \*q;

q=head;

printf("\n The elements are as follows : \n");

while(q!=NULL)

{

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

q=q->next;

}

}

struct node\* insert\_any(struct node\*head)

{

int pos,num,i;

if(head==NULL)

{

printf("\n\nLIST IS EMPTY\n");

return 0;

}

printf("\n Please enter the number that you want to insert : ");

scanf("%d",&num);

printf("\n Please enter the position at which you want to insert an element (NOTE THAT ELEMENT POSITION START FROM 0) ::: ");

scanf("%d",&pos);

struct node \*new,\*q;

new=(struct node \*)malloc(sizeof(struct node));

new->data=num;

new->next=NULL;

if(pos==0)

{

new->next=head;

head=new;

return head;

}

else{

q=head;i=0;

while(i<pos-1 && q!=NULL)

{

q=q->next;

++i;

}

new->next=q->next;

q->next=new;

return head;

}

return head;

}

struct node\* delete\_beg(struct node \*head)

{

struct node \*temp;

temp=head;

head=head->next;

printf("The begining element %d is deleted",temp->data);

free(temp);

return head;

}

struct node\* delete\_end(struct node \*head)

{

struct node \*temp,\*q;

temp=head;

while(temp->next->next!=NULL)

{

temp=temp->next;

}

q=temp->next;

printf("The ending element %d is deleted",q->data);

temp->next=NULL;

free(q);

return head;

}

int main()

{

int ch;

while(1)

{

printf("\n\nThe options are as follows : \n 1. Create linked list \n 2.insert element \n 3. Display element \n 4.Delete from begin \n 5.Delete from end \n6.exit from the code \n enter your choise:");

scanf("%d",&ch);

switch(ch)

{

case 1:head =create\_list(head);break;

//question 1 answer

case 2:head =insert\_any(head);break;

case 3:display(head);break;

//question 2 answer

case 4:head=delete\_beg(head);break;

//question 3 answer

case 5:head=delete\_end(head);break;

case 6:exit(1);

default:printf("\n\n PLEASE ENTER THE CORRECT OPTION\n\n");break;

}

}

}

**Question (4)**

There's no guarantee that beg+end is representable; but in the second case the intermediate values, as well as the expected result, are no larger than end, so there is no danger of overflow.

The second form can also be used for affine types like pointers and other random-access iterators, which can be subtracted to give a distance, but not added together.

**Question (5)**

1. **First, we compare the key with the element at mid1. If found equal, we return mid1.**
2. **If not, then we compare the key with the element at mid2. If found equal, we return mid2.**
3. **If not, then we check whether the key is less than the element at mid1. If yes, then recur to the first part.**
4. **If not, then we check whether the key is greater than the element at mid2. If yes, then recur to the third part.**
5. **If not, then we recur to the second (middle) part.**