**1) To generate a singly-linked list**

// Program to generate a linked list of at least 7 nodes

#include<stdio.h>

#include<stdlib.h>

/\* Creation of a node structure\*/

struct node

{

int data; // Data part

struct node \*next; // Address part

}\*head; // pointer to the first node

void createlist(int n); //function to create list with 'n' nodes

void printlist(); //function to traverse and print the list

int main()

{

int n;

printf("Please enter integer type data while filling node contents\n");

printf("Enter the number(at least 7) of nodes\n");

scanf("%d", &n);

createlist(n);

printf("Contents of the list are:\n");

printlist();

return 0;

}

//function to create list

void createlist(int n)

{

struct node \*newnode, \*temp;

int data, i;

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

//checking for memory allocation

if(head==NULL)

{

printf("Failed to allocate memory");

exit(0);

}

printf("Enter data of node 1:\n");

scanf("%d", &data);

head->data = data; // Linking data field with data

head->next = NULL; // Linking address field to NULL

//Creating the rest n-1 nodes

temp= head; //temporary variable to store address of previous nodes

for(i=2; i<=n; i++)

{

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

//checking availability of memory for new node

if(newnode == NULL)

{

printf("Unable to allocate memory.");

break;

}

printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newnode->data = data; // Linking data field of newNode

newnode->next = NULL; // Make sure new node points to NULL

temp->next = newnode; // Link previous node with newNode

temp = temp->next; // Make current node as previous node

}//end of for loop

}// end of createlist() function

//function to display list

void printlist()

{

struct node \*temp;

// Return if list is empty

if(head == NULL)

{

printf("List is empty.");

return;

}

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data); // Print data of current node

temp = temp->next; // Move to next node

}

}//end of printlist() function

**2) To generate a reverse replica**

//A program illustrating the use of a function to reverse a linearly lined list

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*head;

//functions

void createList(int n);

void reverseList();// to reverse the created list

void displayList();

int main()

{

int n;

printf("Enter the number(at least greater than 7) of nodes:\n ");

scanf("%d", &n);

createList(n);

printf("Here are the contents of the list\n");

displayList();

reverseList();

printf("The list in the reverse order:\n");

displayList();

return 0;

}

void createList(int n)

{

struct node \*newnode, \*temp;

int data, i;

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

//checking for memory allocation

if(head==NULL)

{

printf("Failed to allocate memory");

exit(0);

}

printf("Enter data of node 1:\n");

scanf("%d", &data);

head->data = data; // Linking data field with data

head->next = NULL; // Linking address field to NULL

//Creating the rest n-1 nodes

temp= head; //temporary variable to store address of previous nodes

for(i=2; i<=n; i++)

{

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

//checking availability of memory for new node

if(newnode == NULL)

{

printf("Unable to allocate memory.");

break;

}

printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newnode->data = data; // Linking data field of newNode

newnode->next = NULL; // Make sure new node points to NULL

temp->next = newnode; // Link previous node with newNode

temp = temp->next; // Make current node as previous node

}//end of for loop

}// end of createlist() function

//function to reverse the list

void reverseList()

{

struct node \*prevnode, \*currnode;

if(head != NULL)

{

prevnode = head;

currnode = head->next;

head = head->next;

prevnode->next = NULL; // Make first node as last node

while(head != NULL)

{

head = head->next;

currnode->next = prevnode;

prevnode = currnode;

currnode = head;

}

head = prevnode; // Make last node as head

printf("SUCCESSFULLY REVERSED LIST\n");

}// end of if statement

}//end of reverse function

//function to display list

void displayList()

{

struct node \*temp;

// Return if list is empty

if(head == NULL)

{

printf("List is empty.");

return;

}

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data); // Print data of current node

temp = temp->next; // Move to next node

}

}//end of display list function

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***