PRANJAL PANDEY
201B187 BX-2 B(6)
LAB-7

1.

```
#include <stdio.h> struct
node
{ int
data;
struct node *next;
};
struct node *head, *tail = NULL; void
addAtStart(int data)
{
struct node *newNode = (struct node*)malloc(sizeof(struct
node)); newNode->data = data; newNode->next = NULL;
if(head == NULL)
{
head = newNode; tail
= newNode;
} else
{
struct node *temp = head;
head = newNode; head->next
= temp;
}
}
void display()
```

```
{ struct node *current =
head;
if(head == NULL)
{
printf("List is empty\n");
return; }
printf("Adding nodes to the start of the list: \n");
while(current != NULL)
{ printf("%d ", current-
>data); current = current-
>next;
} printf("\n");
} int main() {
addAtStart(1);
display();
addAtStart(2);
display();
addAtStart(3);
display();
addAtStart(4);
display();
return 0;
} 4.
#include <stdio.h> #include
<stdlib.h>
struct node
{
int data; struct
node *next;
}*head;
```

```
void createList(int n);
void reverseList();
void displayList(); int
main() { int n,
choice;
printf("Enter the total number of nodes: ");
scanf("%d", &n); createList(n);
printf("\nData in the list \n"); displayList();
printf("\nPress 1 to reverse the order of singly linked list\n");
scanf("%d", &choice); if(choice == 1)
{
reverseList();
}
printf("\nData in the list\n");
displayList(); return 0; }
void createList(int n)
{
struct node *newNode, *temp;
int data, i;
if(n <= 0) {
printf("List size must be greater than zero.\n"); return;
}
head = (struct node *)malloc(sizeof(struct node));
if(head == NULL)
{
printf("Unable to allocate memory.");
} else
{
printf("Enter the data of node 1: ");
scanf("%d", &data);
head->data = data;
```

```
head->next = NULL;
temp = head; for(i=2;
i<=n; i++)
{
newNode = (struct node *)malloc(sizeof(struct node));
if(newNode == NULL)
{
printf("Unable to allocate memory.");
break; } else {
printf("Enter the data of node %d: ", i);
scanf("%d", &data);
newNode->data = data;
newNode->next = NULL;
temp->next = newNode; temp
= temp->next;
}
}
printf("SINGLY LINKED LIST CREATED
SUCCESSFULLY\n");
}
}
void reverseList()
{
struct node *prevNode, *curNode; if(head
!= NULL)
{
prevNode = head; curNode
= head->next; head =
head->next;
prevNode->next = NULL; // Make first node as last node
while(head != NULL)
```

```
{
head = head->next; curNode-
>next = prevNode; prevNode
= curNode; curNode = head;
}
head = prevNode; // Make last node as head
printf("SUCCESSFULLY REVERSED LIST\n");
} }
void displayList()
{ struct node
*temp; if(head ==
NULL)
{
printf("List is empty.");
} else { temp = head;
while(temp != NULL)
{
printf("Data = %d\n", temp->data); temp
= temp->next;
}
}
} 5.
#include <stdio.h>
#include <stdlib.h> struct
node
{ int data; struct node
*next; } * head; void
createList(int n); void
displayList(); int
search(int key); int
main() {
```

```
int n, keyToSearch, index;
printf("Enter number of node to create: ");
scanf("%d", &n); createList(n);
printf("\nData in list: \n");
displayList(); printf("\nEnter
element to search: "); scanf("%d",
&keyToSearch); index =
search(keyToSearch);
if (index >= 0)
printf("%d found in the list at position %d\n", keyToSearch,
index + 1); else
printf("%d not found in the list.\n", keyToSearch); return
0;
}
void createList(int n)
{
struct node *newNode, *temp;
int data, i;
head = malloc(sizeof(struct node));
if (head == NULL)
{
printf("Unable to allocate memory. Exiting from app.");
exit(0); }
printf("Enter data of node 1: ");
scanf("%d", &data); head-
>data = data; head->next =
NULL; temp = head; for (i = 2;
i <= n; i++)
{
newNode = malloc(sizeof(struct node));
if (newNode == NULL)
```

```
{
printf("Unable to allocate memory. Exiting from app.");
exit(0); }
printf("Enter data of node %d: ", i);
scanf("%d", &data);
newNode->data = data;
newNode->next = NULL;
temp->next = newNode; temp
= temp->next;
} }
void displayList()
{ struct node
*temp;
if (head == NULL)
{
printf("List is empty.\n");
return; } temp = head;
while (temp != NULL)
{ printf("%d, ", temp-
>data); temp = temp-
>next;
}
printf("\n");
}
int search(int key)
{ int index; struct
node *curNode; index
= 0; curNode = head;
while (curNode != NULL && curNode->data != key)
{ index++;
curNode = curNode->next;
```

```
}
return (curNode != NULL) ? index : -1;
} 6.
#include<stdio.h>
#include<stdlib.h> struct
Node
{
int data; struct
Node* next;
};
void printMiddle(struct Node *head)
{ struct Node *slow_ptr =
head; struct Node *fast_ptr =
head;
if (head!=NULL)
{
while (fast_ptr != NULL && fast_ptr->next != NULL)
{ fast_ptr = fast_ptr->next-
>next; slow_ptr = slow_ptr-
>next;
}
printf("The middle element is [%d]\n\n", slow_ptr->data);
} }
void push(struct Node** head_ref, int new_data)
{
struct Node* new_node =(struct Node*) malloc(sizeof(struct
Node)); new_node->data = new_data; new_node->next =
(*head_ref);
(*head_ref) = new_node;
}
void printList(struct Node *ptr)
```

```
{
while (ptr != NULL)
{
printf("%d->", ptr->data); ptr
= ptr->next;
}
printf("NULL\n");
} int main() { struct Node*
head = NULL;
int i; for (i=5; i>0;
i--) { push(&head,
i); printList(head);
printMiddle(head);
} return 0; }
```