Experiment No.6
Implement Singly Linked List ADT
Name: Shravani Sandeep Raut
Roll No : 48
Date of Performance:
Date of Submission:

Experiment No. 6: Singly Linked List Operations

Aim: Implementation of Singly Linked List

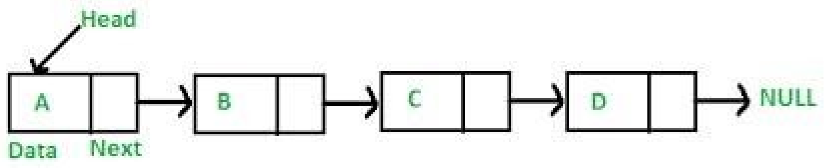
Objective:

It is used to implement stacks and queues which are like fundamental needs throughout computer science. To prevent the collision between the data in the hash map, we use a singly linked list.

Theory:

A linked list is an ordered collection of elements, known as nodes. Each node has two fields: one for data (information) and another to store the address of the next element in the list. The address field of the last node is null, indicating the end of the list. Unlike arrays, linked list elements are not stored in contiguous memory locations; instead, they are connected by explicit links, allowing for dynamic and non-contiguous memory allocation.

The structure of linked list is as shown below



Header is a node containing null in its information field and an next address field contains the address of the first data node in the list. Various operations can be performed on singly linked lists like insertion at front, end, after a given node, before a given node deletion at front, at end and after a given node.

Algorithm

Algorithm to insert a new node at the beginning

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 7 [END OF IF]
```

```
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL NEXT
Step 4: SET DATA = VAL
Step 5: SET NEW_NODE -->NEXT = START
Step 6: SET START = NEW_NODE
Step 7: EXIT
```

Algorithm to insert a new node at the end

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 1 [END OF IF]
Step 2: SET = AVAIL
Step 3: SET AVAIL = AVAIL NEXT
Step 4: SET DATA = VAL
Step 5: SET NEW_NODE = NULL
Step 6: SET PTR = START
Step 7: Repeat Step 8 while PTR NEXT != NULL
Step 8: SET PTR = PTR NEXT [END OF LOOP]
Step 9: SET PTR--> NEXT = New_Node
Step 10: EXIT
```

Algorithm to insert a new node after a node that has value NUM

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 12 [END OF IF]
Step 2: SET = AVAIL
Step 3: SET AVAIL = AVAIL-->NEXT
Step 4: SET DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while != NUM
Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR -->NEXT
[END OF LOOP]
Step 10 : PREPTR--> NEXT = NEW_NODE
Step 11: SET NEW_NODE NEXT = PTR
Step 12: EXIT
```

Algorithm to insert a new node before a node that has value NUM

```
Step 1: IF AVAIL = NULL
        Write OVERFLOW
        Go to Step 12 [END OF IF]
Step 2: SET = AVAIL
Step 3: SET AVAIL = AVAIL-->NEXT
Step 4: SET DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PTR DATA != NUM
Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR -->NEXT
[END OF LOOP]
Step 10: PREPTR-->NEXT = NEW_NODE
Step 11: SET NEXT = PTR
Step 12: EXIT
```

Algorithm to delete the first node

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 5 [END OF IF]
Step 2: SET PTR = START
Step 3: SET START = START -->NEXT
Step 4: FREE PTR
Step 5: EXIT
```

Algorithm to delete the last node

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 8 [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR NEXT != NULL
Step 4: SET PREPTR = PTR
Step 5: SET PTR = PTR -->NEXT [END OF LOOP]
Step 6: SET PREPTR-->NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT
```

Algorithm to delete the node after a given node

```
Step 1: IF START = NULL
```

```
        Write UNDERFLOW
        Go to Step 1 [END OF IF]
Step 2: SET PTR = START
Step 3: SET PREPTR = PTR
Step 4: Repeat Steps 5 and 6 while PREPTR DATA != NUM
Step 5: SET PREPTR = PTR
Step 6: SET PTR = PTR--> NEXT
[END OF LOOP]
Step 7: SET TEMP = PTR
Step 8: SET PREPTR -->NEXT = PTR--> NEXT
Step 9: FREE TEMP
Step 10: EXIT
```

Code:

```
#include <stdio.h>
//#include  <conio.h>
#include  <malloc.h>
struct node
{
int data;
struct node *next;
};

void insertAtBegin(struct node *head)
{
int num;
struct node *temp , *display;
temp = (struct node *)malloc(sizeof(struct node));
printf("\nEnter number: ");  scanf("%d",&num);
temp->data=num;
temp->next=head;
head=temp;
```

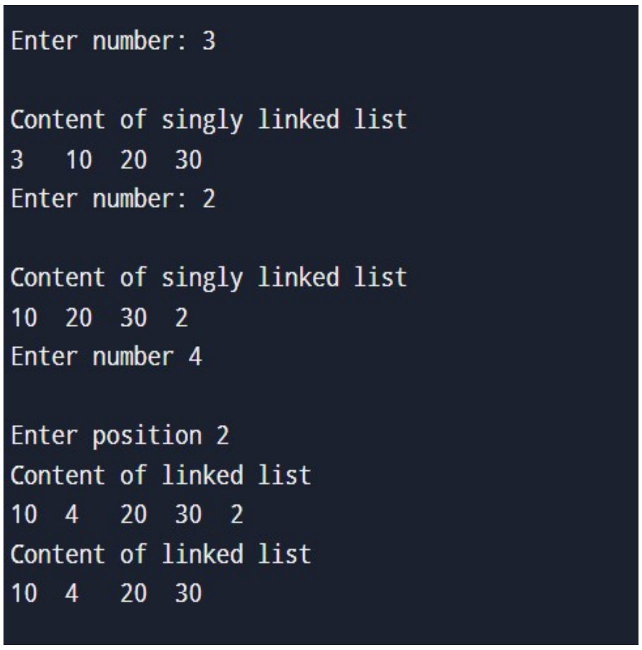
```
display=head;
printf("\nContent of singly linked list\n");
while(display!=NULL)
{
printf("%d\t",display->data);
display=display->next;
}
}

void insertAtEnd(struct node *head)
{
   int num;
   struct node *p = head; struct
   node *temp,*display;
   temp = (struct node *)malloc(sizeof(struct node)); printf("\nEnter
   number: ");
scanf("%d",&num);
temp->data=num; temp-
>next=NULL; while(p-
>next!=NULL)
{
p=p->next;
}
p->next=temp; display=head;
printf("\nContent of singly linked list\n");
while(display!=NULL)
{
printf("%d\t",display->data);
display=display->next;
}
}

void insertAtBet(struct node *head)
{
int num, pos, i;
struct node *temp, *p, *display;
temp = (struct node *)malloc(sizeof(struct node)); printf("\nEnter
number ");
scanf("%d",&num);
temp->data=num;
printf("\nEnter position ");
scanf("%d",&pos);
```

```
p=head;
for(i=1;i<pos-1;i++)
{
p=p->next;
}
temp->next=p->next;
p->next=temp;
display=head;
printf("Content of linked list\n");
while(display!=NULL)
{
printf("%d\t",display->data);
display=display->next;
}

}


void delAtLast(struct node *head)
{

struct node *q, *p, *display;
p=q=head;
while (p->next!=NULL)
{
q=p;
p=p->next;
}
q->next=NULL;
free(p);

display=head;
printf("\nContent of linked list\n");
while(display!=NULL)
{
printf("%d\t",display->data);
display=display->next;
}

}

void delAtStart(struct node *head)
{
```

```
struct node *p,*display;
p=head;
head = head->next;
p->next=NULL;
free(p);
display=head;
printf("\nContent of linked list\n");
while(display!=NULL)
{
printf("%d\t",display->data);
display=display->next;
}

}

void delAtBet(struct node *head)
{

struct node *q, *p, *display;
p=q=head;
int pos,i;
printf("\n Enter position : ");
scanf("%d",&pos);
for(i=1;i<pos;i++)
{
   q=p;
   p=p->next;
}
q->next=p->next;
p->next=NULL;
free(p);

display=head;
printf("\nContent of linked list\n");
while(display!=NULL)
{
printf("%d\t",display->data);
display=display->next;
}

}
```

```
void main()
{
struct node n3; struct
node n2; struct node
n1; struct node
*head ; n3.data = 30;
n2.data = 20;
n1.data = 10;
n3.next = NULL;
n2.next = &n3;
n1.next = &n2; head
= &n1;
//clrscr();

insertAtBegin(head);
insertAtEnd(head);
insertAtBet(head);
delAtLast(head);
delAtStart(head);
delAtBet(head);
//getch();

}
```

Output:



```
Enter number: 3

Content of singly linked list
3   10  20  30
Enter number: 2

Content of singly linked list
10  20  30  2
Enter number 4

Enter position 2
Content of linked list
10  4   20  30  2
Content of linked list
10  4   20  30
```

Conclusion:
<u>Write an example of stack and queue implementation using singly linked list?</u>

Operations: Push the numbers 10, 20, and 30 onto the stack, then pop one element.
stack Operations:
- After pushing: 30 (top), 20, 10
- After popping: 20 (top), 10

Operations: Enqueue the numbers 10, 20, and 30 into the queue, then dequeue one element.
Queue Operations:
- After enqueuing: 10 (front), 20, 30 (rear)
- After dequeuing: 20 (front), 30