SINGLE LINKED LIST

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct node{
        int d;
        struct node *next;
};
struct node *head=NULL;
void insertbeginning(int data){
        struct node *p=(struct node*) malloc(sizeof(struct node));
        p->d=data;
        p->next=head;
        head=p;
}
void insertending(int data){
        struct node *p=(struct node*) malloc(sizeof(struct node));
        p->d=data;
        p->next=NULL;
        if (head == NULL) {
      head = p;
      return;
   }
        struct node *link=head;
        while(link->next!=NULL){
                link=link->next;
        }
        link->next=p;
}
void insertspecificposition(int data,int pos){
        int i;
        struct node *p=(struct node*) malloc(sizeof(struct node));
```

```
            p->d=data;
            struct node *link = head;
            for ( i=1; i<pos-1 && link!=NULL;i++) {
        link=link->next;
    }
            p->next=link->next;
            link->next=p;
}
void display(){
            struct node *temp=head;
            if(temp==NULL){
                        printf("list is empty.\n");
                        return;
    }
    printf("linked list:");
    while(temp!=NULL){
            printf("%d ",temp->d);
            temp=temp->next;
            }
            printf("\n");
}
void deletionbeginning(){
            if(head == NULL){
        printf("List is empty\n");
        return;
    }
            struct node *p=head;
            head=p->next;
            free(p);
}
void deletionend(){
            if(head == NULL){
```

```
        printf("List is empty\n");
        return;
    }
            struct node *p=head;
            struct node *l=NULL;
            while(p->next!=NULL){
                    l=p;
                    p=p->next;
            }
            l->next=NULL;
            free(p);
}
void deletionsp(int pos){
            int i;
            struct node *p=head;
            struct node *l=NULL;
            if(pos==1){
                    head=head->next;
                    free(p);
            }
            for(i=1;i<pos&&p!=NULL;i++){
                    l=p;
                    p=p->next;
            }
            free(p);
}
int main(){
            int choice,data,position,index;
            while(1){
        printf("1.Insert at Beginning\n");
        printf("2.Insert at End\n");
        printf("3.Insert at Specific Position\n");
```

```
printf("4.Display\n");
printf("5.Deletion at Beginning\n");
printf("6.Deletion at End\n");
printf("7.Deletion at Specific Position\n");
printf("8.Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch(choice){
    case 1:
            printf("enter the data:");
            scanf("%d",&data);
            insertbeginning(data);
            break;
    case 2:
      printf("enter data:");
      scanf("%d",&data);
      insertending(data);
      break;
   case 3:
      printf("enter data:");
      scanf("%d", &data);
      printf("enter the position:");
      scanf("%d",&position);
      insertspecificposition(data,position);
      break;
   case 4:
      display();
      break;
   case 5:
    deletionbeginning();
    break;
   case 6:
```

```
            deletionend();
            break;
           case 7:
            printf("enter the position:");
            scanf("%d",&index);
            deletionsp(index);
            break;
           case 8:
              printf("exiting...\n");
              exit(0);
           default:
              printf("invalid choice!\n");
                    }
            }
}
```

OUTPUT:S