

```
#include <stdio.h>
```

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node *link;
```

```
}
```

```
typedef struct node *NODE;
```

```
NODE getnode()
```

```
{
```

```
    NODE x;
```

```
    x = (NODE) sizeof malloc (sizeof(NODE));
```

```
    if (x == NULL)
```

```
    {
```

```
        printf("Memory full\n");
```

```
        exit(10);
```

```
    }
```

```
    return x;
```

```
}
```

```
void freenode(NODE x)
```

```
{ free(x); }
```

```
NODE push(NODE first, int item)
```

```
{
```

```
    NODE temp;
```

```
    temp = getnode();
```

```
    temp->info = item;
```

```
    temp->link = NULL;
```

```
    if (first == NULL)
```

```
        return temp;
```

```
    temp->link = first;
```

```
first = temp;  
return first;
```

```
int  
NODE pop(NODE first)
```

```
{  
    NODE temp;  
    if (first == NULL)
```

```
{  
        printf("Stack underflow\n");  
        return first;
```

```
}  
temp = first;
```

```
temp = temp->link;
```

```
printf("item deleted. The first end is = %d\n", first->info);  
free(first);
```

```
return temp;
```

```
}  
void display()
```

```
{  
    NODE temp;
```

```
if (first == NULL)
```

```
{  
        printf("Stack empty\n");
```

```
for (temp = first; temp != NULL; temp = temp->link)
```

```
{  
        printf("%d\n", temp->info);
```

```
}  
int main()
```

```
{  
    int item, choice;
```

```
    NODE first = NULL;
```

```
    for(;;)
```

```
{  
        printf("\n 1. Push  \n 2. Pop  \n 3. display  \n 4. Exit  \n");
```



```
printf("Enter your choice \n");
scanf("%d \n", &choice);
switch(choice)
```

```
{
    case 1: printf("Enter the item \n");
            scanf("%d \n", &item);
            first = push(first, item);
            break;
```

```
    case 2: first = pop(first);
            break;
```

```
    case 3: display();
            break;
```

```
    case 4: default: exit(0);
            break;
```

```
    }
}
}
```

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 struct node {
4     int info;
5     struct node*link;
6 };
7 typedef struct node*NODE;
8 NODE getnode(){
9     NODE x;
10    x=(NODE)malloc(sizeof(struct node));
11    if(x==NULL) {
12        printf("memfull\n");
13        exit(0);
14    }
15    return x;
16 }
17 void freenode(NODE x){
18     free(x);
19 }
20 NODE insert_front(NODE first,int item) {
21     NODE temp;
22     temp=getnode();
23     temp->info=item;
24     temp->link=NULL;
25     if(first==NULL)
26         return temp;
27     temp->link=first;
28     first=temp;
29     return first;
30 }
31 NODE delete_front(NODE first){
32     NODE temp;
33     if(first==NULL) {
34         printf("stack is empty cannot delete\n");
35         return first;
36     }
37     temp=first;

```

```

38     temp=temp->link;
39     printf("item deleted at front-end
is=%d\n",first->info);
40     free(first);
41     return temp;
42 }
43 void display(NODE first) {
44     NODE temp;
45     if(first==NULL)
46         printf("stack empty cannot display
items\n");
47     for(temp=first;temp!=NULL;temp=temp->link)
48     {
49         printf("%d\n",temp->info);
50     }
51 int main() {
52     int item,choice;
53     NODE first=NULL;
54     for(;;) {
55
56         printf("\n1:Insert_front\n2:Delete_front\n3:D
isplay_list\n4:Exit\n");
57         printf("enter the choice\n");
58         scanf("%d",&choice);
59         switch(choice) {
60             case 1:printf("enter the item at
front-end\n");
61                 scanf("%d",&item);
62                 first=insert_front(first,item);
63                 break;
64             case 2:first=delete_front(first);
65                 break;
66             case 3:display(first);
67                 break;
68             default:exit(0);
69         }
70     }

```

```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
10

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
20

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
30

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
30
20
10

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30

1:Insert_front
2:Delete_front
3:Display_list
```



```
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=20
```

```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=10
```

```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
stack is empty cannot delete
```

```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
stack empty cannot display items
```

```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
```