main.c

```c
/*stack using sll*/

#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
      printf("mem full\n");
      exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_front(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;
}
NODE delete_front(NODE first)
{
```

main.c

```
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("stack is empty cannot delete\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("item deleted at front-end is=%d\n",first->info);
    free(first);
    return temp;
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    printf("stack empty cannot display items\n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
       printf("%d\n",temp->info);
    }
}
int main()
{
    int item,choice,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("\n 1:Insert_front\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        printf("-----\n");
        switch(choice)
        {
            case 1:printf("enter the item at front-end\n");
            scanf("%d",&item);
```

main.c

```
49      free(first);
50      return temp;
51  }
52  void display(NODE first)
53  {
54      NODE temp;
55      if(first==NULL)
56      printf("stack empty cannot display items\n");
57      for(temp=first;temp!=NULL;temp=temp->link)
58      {
59          printf("%d\n",temp->info);
60      }
61  }
62  int main()
63  {
64      int item,choice,pos;
65      NODE first=NULL;
66      for(;;)
67      {
68          printf("\n 1:Insert_front\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
69          printf("enter the choice\n");
70          scanf("%d",&choice);
71          printf("-----\n");
72          switch(choice)
73          {
74              case 1:printf("enter the item at front-end\n");
75              scanf("%d",&item);
76              first=insert_front(first,item);
77              break;
78              case 2:first=delete_front(first);
79              break;
80              case 3:display(first);
81              break;
82              default:exit(0);
83              break;
84          }
85      }
86  }
87
```

input

```
 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
-----
enter the item at front-end
23

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
-----
enter the item at front-end
45

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
-----
enter the item at front-end
67

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

input

```
enter the choice
3
-----
67
45
23

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
-----
item deleted at front-end is=67

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
-----
item deleted at front-end is=45

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
-----
23

 1:Insert_front
```

```
3
-----
23

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
-----
item deleted at front-end is=23

 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
```

main.c

```c
/*implement queue using Linked list*/

#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("mem full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_rear(NODE first,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
    return temp;
    cur=first;
    while(cur->link!=NULL)
    cur=cur->link;
    cur->link=temp;
    return first;
```

main.c

```
    cur->link=temp;
    return first;
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("list is empty cannot delete\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("item deleted at front-end is=%d\n",first->info);
    free(first);
    return temp;
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    printf("list empty cannot display items\n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
}
int main()
{
    int item,choice,pos;
    NODE first=NULL;

    for(;;)
    {
        printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        printf("-----\n");
        switch(choice)
```

main.c

```c
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    printf("list empty cannot display items\n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
}
int main()
{
    int item,choice,pos;
    NODE first=NULL;

    for(;;)
    {
        printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        printf("-----\n");
        switch(choice)
        {
            case 1:printf("enter the item at rear-end\n");
            scanf("%d",&item);
            first=insert_rear(first,item);
            break;
            case 2:first=delete_front(first);
            break;
            case 3:display(first);
            break;
            default:exit(0);
            break;
        }
    }

}
```

input

```
 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
-----
enter the item at rear-end
23

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
-----
enter the item at rear-end
45

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
-----
enter the item at rear-end
78

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

```
 3:Display_list
 4:Exit
enter the choice
3
-----
23
45
78

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
-----
item deleted at front-end is=23

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
-----
item deleted at front-end is=45

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
-----
78
```