1. WAP to using dynamic implementation of linked list.

```
#include<stdio.h>
#include<conio.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;
}
NODE deletefront (NODE first)
{
  NODE temp;
  if (first == NULL)
{
  printf ("list is empty can't delete \n");
  return first;
}
  temp = first;
  temp = temp->link;
  printf ("item deleted at front end: %d", first
  free (first);
  return temp;
}

void display (NODE first)
{
  NODE temp;
  if (first == NULL)
printf ("list empty cannot display \n");
  for ( temp = first; temp != NULL; temp = temp
  {
    printf ("%d\n", temp->info);
}
}
```

```
void main()
{
  int item, choice, pos;
  NODE first = NULL;
  for (;;)
{
printf ("\n1. insert_rear\n2. delete_front\n
        3. display list\n4: Exit\n");
printf ("enter the choice\n");
scanf ("%d", &choice);
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;
   }
}
getch();
}
```

Q2. WAP to implement stacks using Linked List

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.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 insertfront(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)
{
    NODE temp;
    if (first == NULL)
    {
    printf ("stack is empty can't 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 can't display\n");
    for (temp = first; temp != NULL; temp=
                                temp->link)
    {
    printf ("%d\n", temp->info);
}
}
```

```
void main()
{
 int item, choice, pos;
 NODE first = NULL;
 for(;;)
{ printf("\n1.insertfront[PUSH]\n2.delete
      front[pop]\n3.Display_list\n
      4:Exit\n");
 printf("enter the choice\n");
 scanf("%d",&choice);
 switch(choice)
   {
   case 1: printf("enter item at
           front end\n");
           scanf("%d",&item);
           first = insert_front(first,item);
           break;
   case 2 : first = deletefront(first);
          break;
   case 3 : display(first);
          break;
   default : exit(0);
          break;
 }
}
```