|  |  |
| --- | --- |
| 1 | Write a program to insert and delete element at any position of a double link list. |
|  | #pragma pack(1)  struct xxx  {  struct xxx \*left;  int roll;  struct xxx \*right;  };  struct xxx \*p,\*q,\*m;  main()  {  create\_double();  insert(m,3);  delete(m,3);  traverse\_rtol(p);  printf("\n");  traverse\_ltor(m);    }  int insert(struct xxx \*p,int pos)  {  struct xxx \*r,\*q;  int i;  for(i=1;i<pos;i++)  {  r=p;  p=p->right;  }  q=malloc(sizeof(struct xxx));  printf("Enter roll:");  scanf("%d",&q->roll);  r->right=q;  q->left=r;  q->right=p;  p->left=q;  }  int delete(struct xxx \*p,int pos)  {  int i;  struct xxx \*r;  for(i=1;i<pos;i++)  {  r=p;  p=p->right;  }  r->right=p->right;  p->right->left=r;  free(p);  }  int traverse\_rtol(struct xxx \*p)  {  while(p!=0)  {  printf("%d ",p->roll);  p=p->left;  }  }  int traverse\_ltor(struct xxx \*p)  {  while(p!=0)  {  printf("%d ",p->roll);  p=p->right;  }  }  int create\_double()  {  char ch[10];  p=malloc(sizeof(struct xxx));  p->left=0;  m=p;  printf("Enter roll:");  scanf("%d",&p->roll);  while(1)  {  printf("Do u continue yes/no:");  scanf("%s",ch);  if(strcmp(ch,"no")==0)  break;  q=malloc(sizeof(struct xxx));  p->right=q;  q->left=p;  p=q;  printf("Enter roll:");  scanf("%d",&p->roll);  }  p->right=0;  } |
| 2 | Implement binary search in a linked list. |
|  | int binary\_search(struct xxx \*p)  {  int n;  struct xxx \*m,\*l=p,\*u=0;  printf("Enter number to search:");  scanf("%d",&n);    while(l!=u)  {  m=find\_middle(l,u);  if(n==m->roll)  {  printf("Found");  return;  }  else  if(n>m->roll)  l=m->ad;  else  u=traverse\_lower(p,m);    }  printf("not found");  } |
| 3 | Implement linear search in a linked list |
|  | int linear\_search(struct xxx \*p)  {  int n;  printf("Enter number to search:");  scanf("%d",&n);  while(p!=0)  {  if(n==p->roll)  {  printf("Found");  return;  }  p=p->ad;  }  printf("Not found");  } |
| 4 | Interchange each 2 alternate nodes of a linked list |
|  | #pragma pack(1)  struct xxx  {  int roll;  struct xxx \*ad;  };  struct xxx \*create\_list();  main()  {  struct xxx \*b;    b=create\_list();  interchange(b);  //traverse(b);    }  int interchange(struct xxx \*p)  {  struct xxx \*q=p->ad;  int temp;  while(q!=0)  {  temp=p->roll;  p->roll=q->roll;  q->roll=temp;  if(q->ad==0 || p->ad->ad==0 )  break;  p=p->ad->ad;  q=q->ad->ad;  }  } |