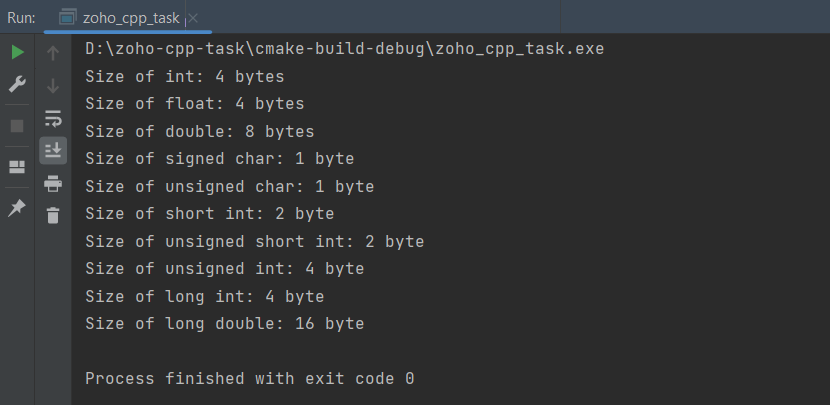
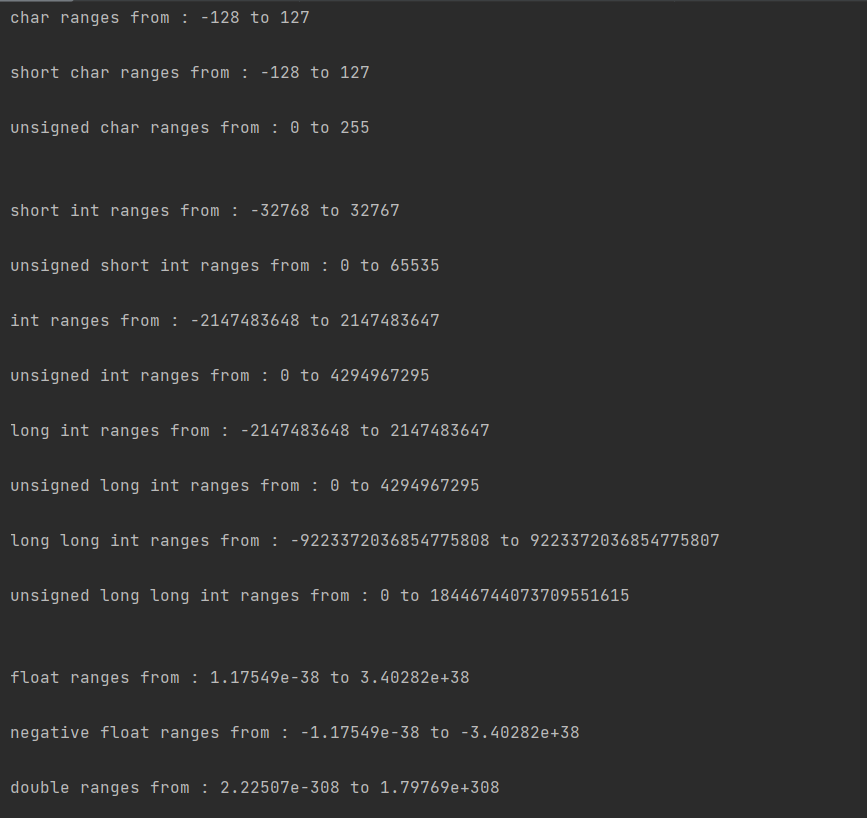
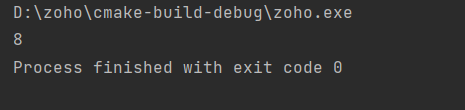
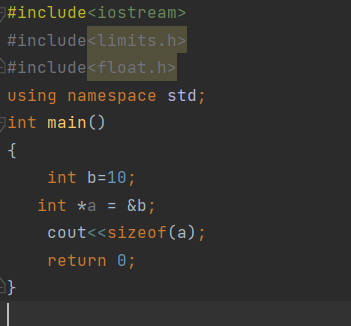
#include <iostream>  
  
int main() {  
  
 short int si;  
 unsigned short int usi;  
 unsigned int ui;  
 int i;  
 long int li;  
 signed char sc;  
 unsigned char uc;  
 float f;  
 double d;  
 long double ld;  
  
  
 printf("Size of int: %ld bytes\n",sizeof(i));  
 printf("Size of float: %ld bytes\n",sizeof(f));  
 printf("Size of double: %ld bytes\n",sizeof(d));  
 printf("Size of signed char: %ld byte\n",sizeof(sc));  
 printf("Size of unsigned char: %ld byte\n",sizeof(uc));  
 printf("Size of short int: %ld byte\n",sizeof(si));  
 printf("Size of unsigned short int: %ld byte\n",sizeof(usi));  
 printf("Size of unsigned int: %ld byte\n",sizeof(ui));  
 printf("Size of long int: %ld byte\n",sizeof(li));  
 printf("Size of long double: %ld byte\n",sizeof(ld));  
 return 0;  
}



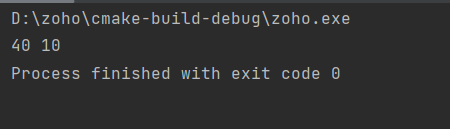
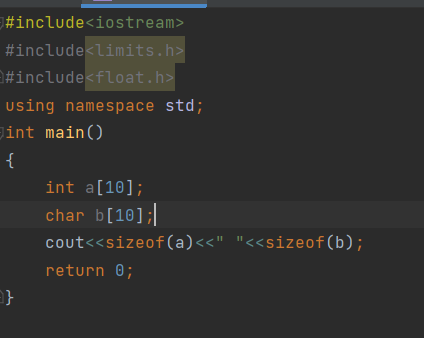
Minimum and maximum ranges:

#include<iostream>  
#include<limits.h>  
#include<float.h>  
using namespace std;  
int main()  
{  
   
 cout << "char ranges from : " << CHAR\_MIN << " to " << CHAR\_MAX;  
 cout << "\n\nshort char ranges from : " << SCHAR\_MIN << " to " << SCHAR\_MAX;  
 cout << "\n\nunsigned char ranges from : " << 0 << " to " << UCHAR\_MAX;  
 cout << "\n\n\nshort int ranges from : " << SHRT\_MIN << " to " << SHRT\_MAX;  
 cout << "\n\nunsigned short int ranges from : " << 0 << " to " << USHRT\_MAX;  
 cout << "\n\nint ranges from : " << INT\_MIN << " to " << INT\_MAX;  
 cout << "\n\nunsigned int ranges from : " << 0 << " to " << UINT\_MAX;  
 cout << "\n\nlong int ranges from : " << LONG\_MIN << " to " << LONG\_MAX;  
 cout << "\n\nunsigned long int ranges from : " << 0 << " to " << ULONG\_MAX;  
 cout << "\n\nlong long int ranges from : " << LLONG\_MIN << " to " << LLONG\_MAX;  
 cout << "\n\nunsigned long long int ranges from : " << 0 << " to " << ULLONG\_MAX;  
 cout << "\n\n\nfloat ranges from : " << FLT\_MIN << " to " << FLT\_MAX;  
 cout << "\n\nnegative float ranges from : " << -FLT\_MIN << " to " << -FLT\_MAX;  
 cout << "\n\ndouble ranges from : " << DBL\_MIN << " to " << DBL\_MAX;  
 cout << "\n\nnegative double ranges from : " << -DBL\_MIN << " to " << +DBL\_MAX;  
  
 return 0;  
  
}



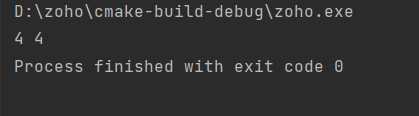
Size of pointer always remains 8 bytes for any data type

Size of integer array will be 4 time the character array for same number of elements because int is 4 bytes and char is 1 byte



The data type size and their order of arrangement decides size class or struct

#include<iostream>  
using namespace std;  
struct node{  
 int a;  
};  
class node1{  
 int a;  
};  
int main()  
{  
 node a;node1 b;  
 cout<<sizeof(a)<<" "<<sizeof(b);  
 return 0;  
}



**Different types of memory allocation:**

Static Memory allocation:

Memory allocated during starting of program.cannot be resized or changed after allocation this is the drawback.

Dynamic Memory allocation:

calloc():

cptr = (int\*) calloc(35, sizeof (int));

It initializes memory bits to 0 defaulty

malloc():

cptr = (int\*) calloc(35, sizeof (int));

it do not initialize bits so allocated memory will have garbage values

it will return NULL if no memory is available

free():

free(cptr);

calloc and malloc just initializes the memory and cannot release on its own,

so we need to explicitly release the memory using free to avoid memory leakage

realloc():

realloc(cptr,n);

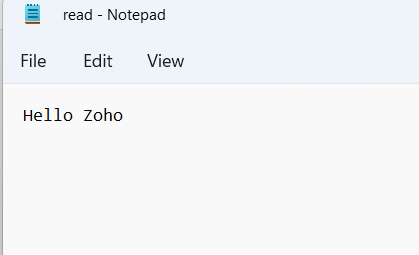
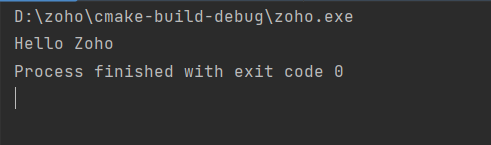
to change allocated memory size

**Is sizeof all dataType decided by compiler or OS?**

the sizeof data type is decided by compiler but hardware and OS also plays a role in deciding i.e because of processor architecture 32bit or 64 bit.

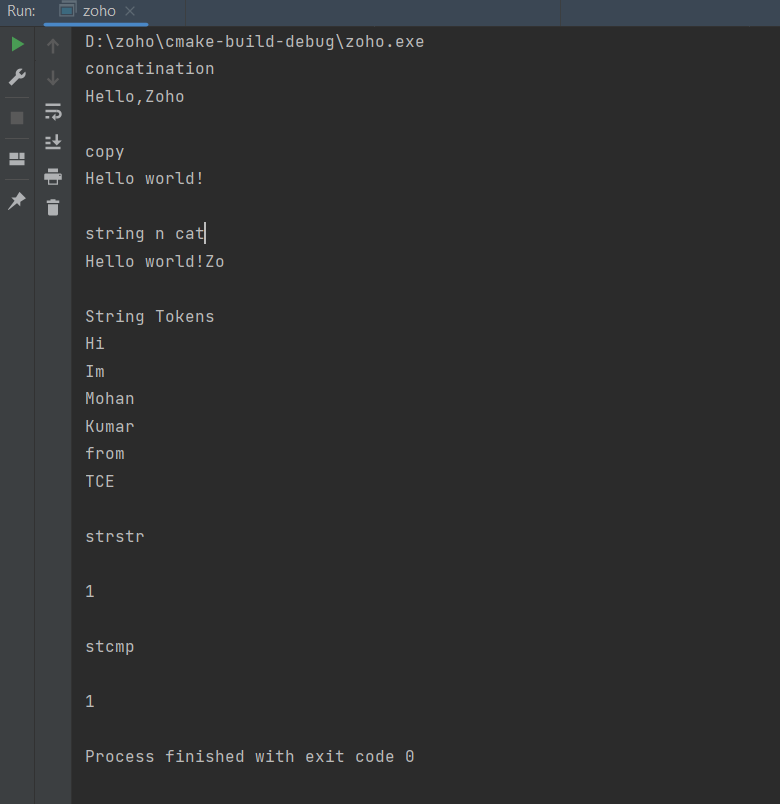
File reading with dynamic memory allocation:

#include<stdio.h>  
#include<stdlib.h>  
  
#define SIZE 1024  
  
int main()  
{  
 FILE \*fp;  
 char \*buf = (char\*)malloc(SIZE);  
 size\_t nread;  
  
 fp = fopen("D:/read.txt", "r");  
 if(fp)  
 {  
 while(fgets(buf, SIZE, fp) != NULL)  
 {  
 printf("%s", buf);  
 }  
 }  
 fclose(fp);  
 return 0;  
}



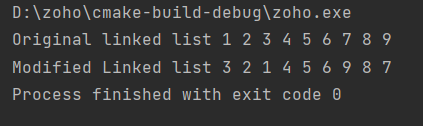
**C functions Own Implementations**

#include <iostream>  
#include <string.h>  
using namespace std;  
  
void str\_cat(char des[],char src[]){  
 int i = 0, j = 0;  
 while (des[i] != '\0')  
 i += 1;  
 while (src[j] != '\0') {  
 des[i + j] = src[j];  
 j += 1;  
 }  
 des[i + j] = '\0';  
  
 printf("%s\n", des);  
}  
void str\_cpy(char des[],char src[]){  
 int i=0;  
 while(src[i] != '\0'){  
 des[i] = src[i];i++;  
 }  
 des[i] = '\0';  
 printf("%s\n",des);  
}  
void str\_ncat(char des[],char src[],int n)  
{  
 int i = 0, j = 0;  
 while (des[i] != '\0')  
 i ++;  
 while (src[j] != '\0' && j<n) {  
 des[i + j] = src[j];  
 j ++;  
 }  
 des[i + j] = '\0';  
  
 printf("%s\n", des);  
}  
  
char\* str\_tok(char\* s, char d)  
{  
 static char\* input = NULL;  
 if (s != NULL)  
 input = s;  
 if (input == NULL)  
 return NULL;  
 char\* result = new char[strlen(input) + 1];  
 int i = 0;  
 for (; input[i] != '\0'; i++) {  
 if (input[i] != d)  
 result[i] = input[i];  
 else {  
 result[i] = '\0';  
 input = input + i + 1;  
 return result;  
 }  
 }  
 result[i] = '\0';  
 input = NULL;  
 return result;  
}  
  
  
int compare(const char \*X, const char \*Y)  
{  
 while (\*X!='\0' && \*Y!='\0')  
 {  
 if (\*X != \*Y) {  
 return 0;  
 }  
 X++;  
 Y++;  
 }  
 return 1;  
}  
  
bool str\_str(const char\* X, const char\* Y)  
{  
 while (\*X != '\0')  
 {  
 if ((\*X == \*Y) && compare(X, Y)) {  
 return true;  
 }  
 X++;  
 }  
 return false;  
}bool str\_cmp(const char \*a,const char \*b){  
 if(\*a=='\0' && \*b=='\0')return true;  
 else if(\*a==\*b){  
 return str\_cmp(a+1,b+1);  
 }  
 return false;  
  
}  
int main() {  
  
 char des[1000]="Hello,", src[100]="Zoho";  
 cout<<"concatination\n";  
 str\_cat(des,src);  
 char copy[20] = "Hello world!";  
 cout<<"\ncopy\n";  
 str\_cpy(des,copy);  
 cout<<"\nstring n cat\n";  
 str\_ncat(des,src,2);  
 cout<<"\nString Tokens\n";  
 char forTokens[50] = "Hi Im Mohan Kumar from TCE";  
 char\* ptr = str\_tok(forTokens, ' ');  
 cout << ptr << endl;  
 while (ptr != NULL) {  
 ptr = str\_tok(NULL, ' ');  
 cout << ptr << endl;  
 }  
  
  
 printf("\nstrstr\n");  
 char \*a = "hello zoho world",\*b = "zoho";  
 printf("\n%d\n", str\_str(a, b));  
 printf("\nstcmp\n");  
 char \*c = "zoho",\*d = "zoho";  
 printf("\n%d\n", str\_cmp(c, d));  
 return 0;  
}



**K Alternate reversing of Linked List in O(N) time complexity**

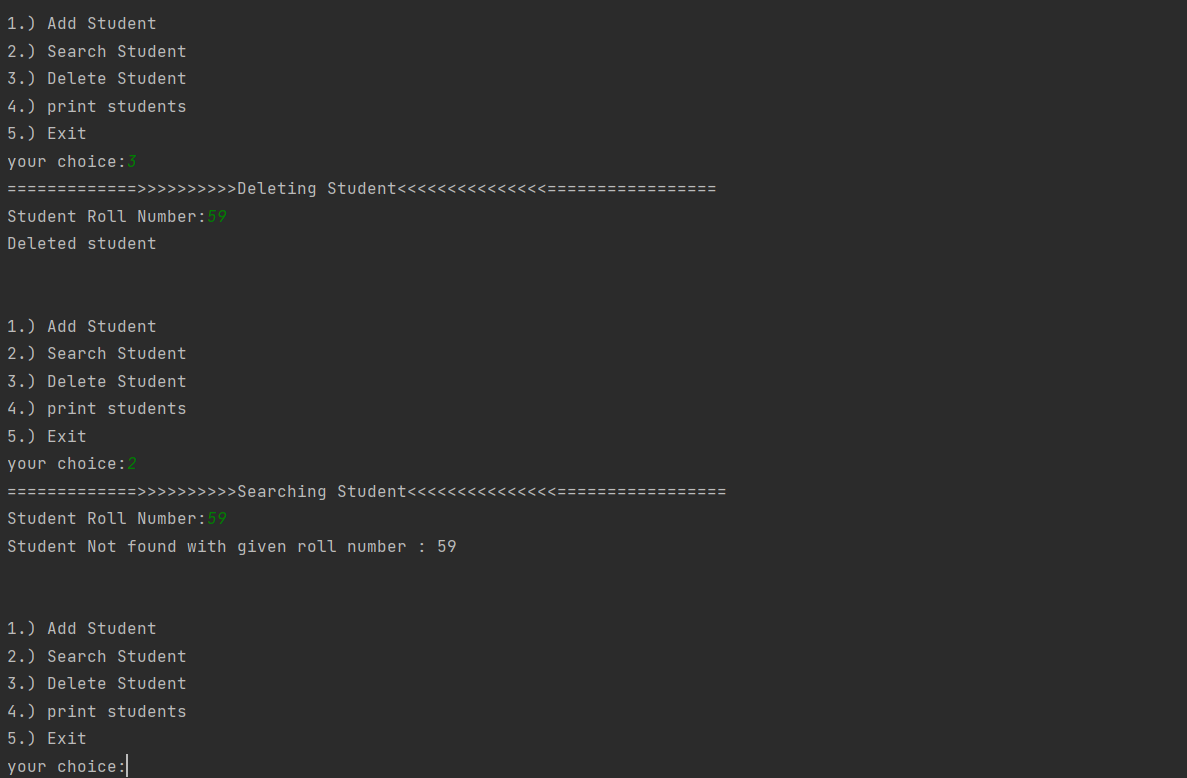
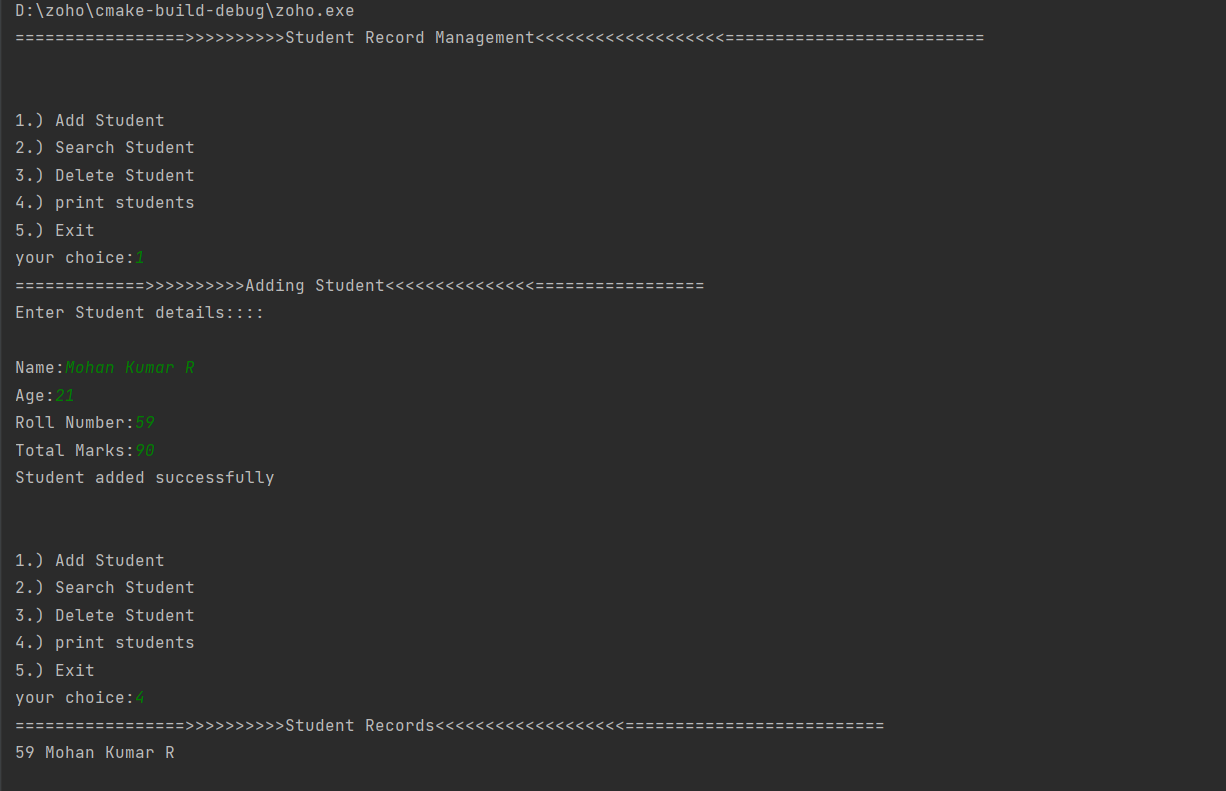
#include <bits/stdc++.h>  
using namespace std;  
class Node  
{  
public:  
 int data;  
 Node\* next;  
 Node(int data){  
 this->data = data;  
 }  
};  
  
Node \*kaltrev(Node \*head, int k)  
{  
 Node\* curr = head;  
 Node\* next;  
 Node\* prev = NULL;  
  
 int count = 0; while (curr != NULL && count < k)  
 {  
 next = curr->next;  
 curr->next = prev;  
 prev = curr;  
 curr = next;  
 count++;  
 }  
  
 if(head != NULL)  
 head->next = curr;  
 count = 0;  
 while(count < k-1 && curr != NULL )  
 {  
 curr = curr->next;  
 count++;  
 }  
 if(curr != NULL)  
 curr->next = kaltrev(curr->next, k);  
 return prev;  
}  
  
void printList(Node \*node)  
{  
 int count = 0;  
 while(node != NULL)  
 {  
 cout<<node->data<<" ";  
 node = node->next;  
 count++;  
 }  
}  
  
int main(void)  
{  
  
 Node\* head = new Node(1);  
 int i;  
 Node\* temp = head;  
 for(i = 2; i<10; i++)  
 {  
 temp->next = new Node(i);  
 temp = temp->next;  
 }  
 temp->next = NULL;  
 cout << "Original linked list ";  
 printList(head);  
 head = kaltrev(head, 3);  
 cout<<endl;  
 cout << "Modified Linked list ";  
 printList(head);  
 return(0);  
}



**Student record storing:**

#include <bits/stdc++.h>  
using namespace std;  
  
struct Student {  
 string name;  
 int roll\_number;  
 int age;  
 double total\_marks;  
 Student \*next;  
};  
Student \*head,\*tail;  
void add(){  
 cout<<"=============>>>>>>>>>>Adding Student<<<<<<<<<<<<<<<=================\n";  
 Student \*student = new Student();  
 student->next = NULL;  
 cout<<"Enter Student details::::\n\n";  
 cin.ignore();  
 cout<<"Name:";getline(cin,student->name);  
 cout<<"Age:";cin>>student->age;  
 cout<<"Roll Number:";cin>>student->roll\_number;  
 cout<<"Total Marks:";cin>>student->total\_marks;  
 cout<<"Student added successfully\n";  
 if(head==NULL){  
 head = student;  
 tail = student;  
 return;  
 }  
 tail->next = student;  
 tail = tail->next;  
}  
void search(){  
 Student \*trav = head;  
 cout<<"=============>>>>>>>>>>Searching Student<<<<<<<<<<<<<<<=================\n";  
 cout<<"Student Roll Number:";  
 int roll;cin>>roll;  
 while(trav!=NULL){  
 if(trav->roll\_number==roll){  
 cout<<"Student Details::::\n";  
 cout<<"Name:"<< trav->name<<endl;  
 cout<<"Age:"<<trav->age<<endl;  
 cout<<"Total Marks:"<<trav->total\_marks;  
  
 return;  
 }  
 trav = trav->next;  
 }  
 cout<<"Student Not found with given roll number : "<<roll<<endl;  
}  
  
void Delete(){  
 Student \*trav = head,\*prev = NULL;  
 cout<<"=============>>>>>>>>>>Deleting Student<<<<<<<<<<<<<<<=================\n";  
 cout<<"Student Roll Number:";  
 int roll;cin>>roll;  
 while(trav){  
 if(trav->roll\_number==roll){  
 if(prev==NULL){  
 head = head->next;  
 free(trav);  
 }  
 else{  
 prev->next = trav->next;  
 free(trav);  
 }  
 cout<<"Deleted student\n";  
 return;  
 }  
 prev = trav;  
 trav = trav->next;  
 }  
 cout<<"Student Not found with given roll number : "<<roll<<endl;  
}  
void printStudents(){  
 Student \*trav=head;  
 cout<<"=================>>>>>>>>>>Student Records<<<<<<<<<<<<<<<<<<<==========================\n";  
 while(trav){  
 cout<<trav->roll\_number<<" "<<trav->name<<endl;  
 trav=trav->next;  
 }  
}  
int main(void)  
{  
 cout<<"=================>>>>>>>>>>Student Record Management<<<<<<<<<<<<<<<<<<<==========================\n";  
 bool exit =false;  
 while(!exit){  
 cout<<endl;cout<<endl;  
 int choice;  
 cout<<"1.) Add Student\n2.) Search Student\n3.) Delete Student\n4.) print students\n5.) Exit\nyour choice:";  
 cin>>choice;  
 switch(choice){  
 case 1:  
 add();  
 break;  
 case 2:  
 search();  
 break;  
 case 3:  
 Delete();  
 break;  
 case 4:  
 printStudents();  
 break;  
 case 5:  
 exit = true;

Break;  
 default:  
 cout<<"Wrong Choice------:(\n";  
 }  
 }  
 return(0);  
}

****